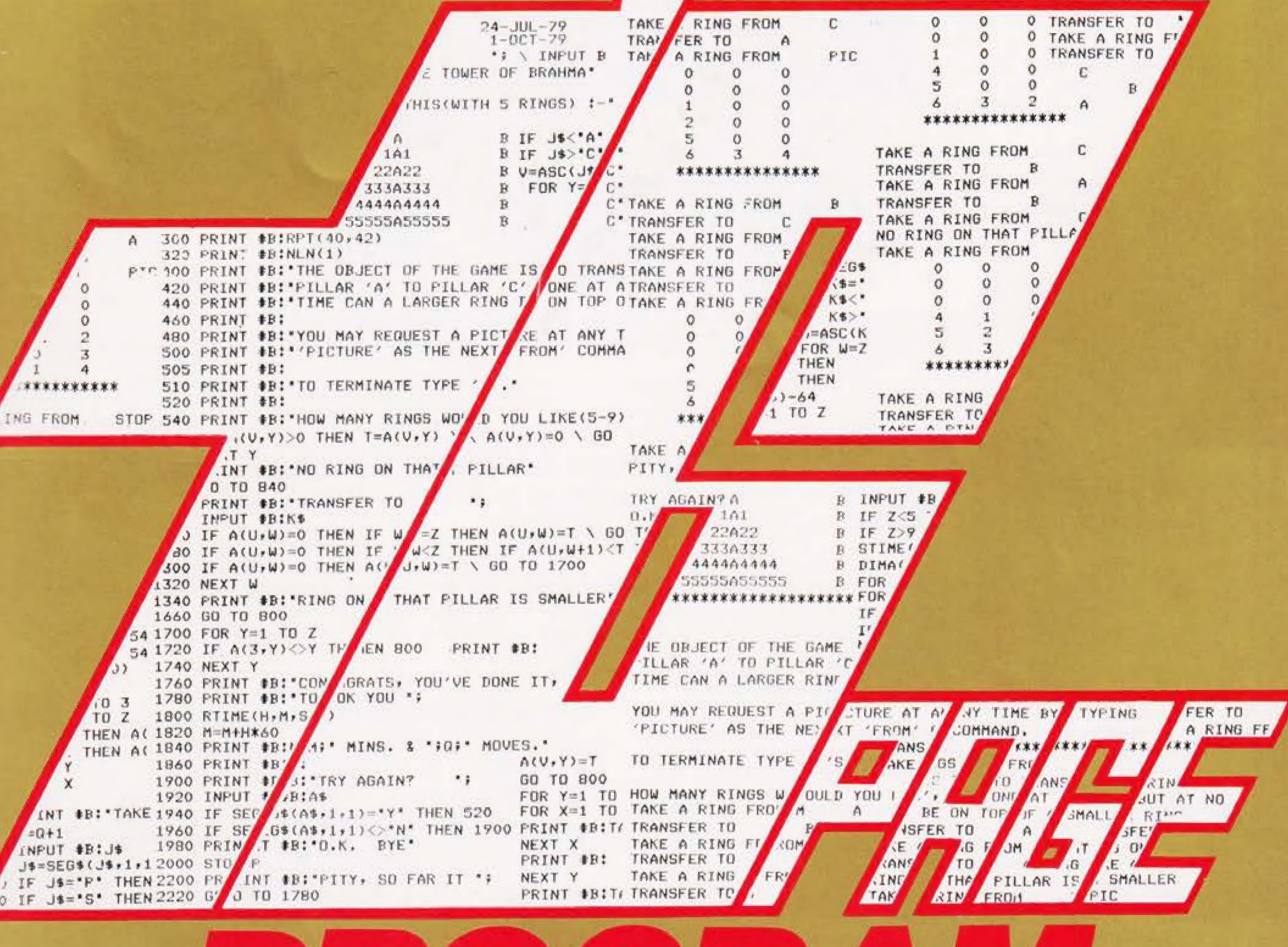
computing today

ISSN 0142-7210
APR 1980
60p



16 PAGE PROGRAM SUPPLEMENT

PROGRAMMERS SPECIAL ISSUE

- Machine Code
- Program Library
- Pico~Basic
- Pet Bits

8K ON BOARD MEMORY!
5K RAM, 3K ROM or 4K RAM, 4K ROM (link selectable). Kit supplied with 3K RAM, 3K ROM. System expandable for up to 32K memory.

2 KEYBOARDS!
56 Key alphanumeric keyboard for entering high level language plus 16 key Hex pad for easy entry of machine code.

GRAPHICS!
64 character graphics option — includes transistor symbols! Only £18.20 extra!

MEMORY MAPPED
high resolution VDU circuitry using discrete TTL for extra flexibility. Has its own 2K memory to give 32 lines for 64 characters.

KANSAS CITY
low error rate tape interface

NEW LOW PRICE!

Cabinet size 19.0" × 15.7" × 3.3". Television by courtesy of Rumblelows Ltd., price £58.62

2 MICROPROCESSORS
Z80 the powerful CPU with 158 instruction, including all 78 of the 8080, controls the MM57109 number cruncher. Functions include +, —, *, /, squares, roots, logs, exponentials, trig functions, inverses etc. Range 10^{-99} to 9×19^{99} to 8 figures plus 2 exponent digits.

EFFICIENT OPERATION
Why waste valuable memory on sub routines for numeric processing? The number cruncher handles everything internally!

RESIDENT BASIC
with extended mathematical capability. Only 2K memory used but more powerful than most 8K Basics!

1K MONITOR
resident in EPROM

SINGLE BOARD DESIGN
Even keyboards and power supply circuitry on the superb quality double sided plated through-hole PCB.

COMPLETE KIT NOW ONLY £249 + VAT

Kit also available as separate packs: e.g. PCB, Keyboards, Cabinet, etc.

POWERTRAN

PSI Comp 80.Z80 Based powerful scientific computer Design as published in Wireless World

The kit for this outstandingly practical design by John Adams published in a series of articles in Wireless World really is complete!

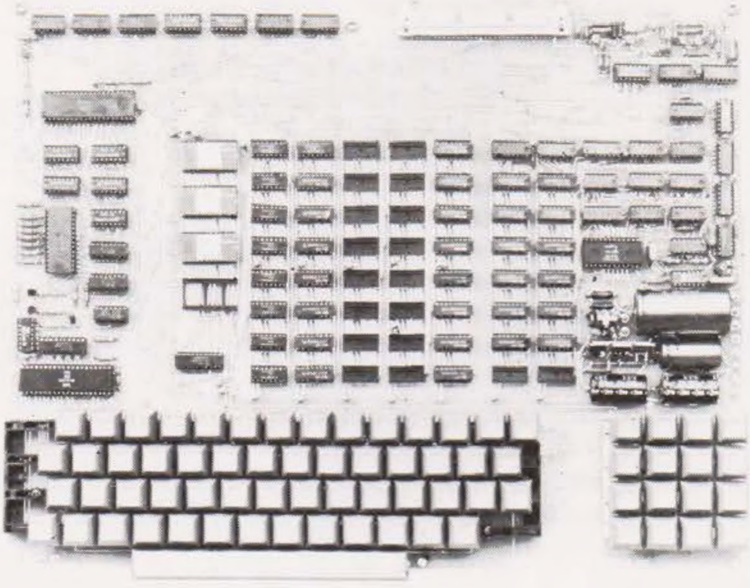
Included in the PSI COMP 80 scientific computer kit is a professionally finished cabinet, fibre-glass double sided, plated-through-hole printed circuit board, 2 keyboards PCB mounted for ease of construction, IC sockets, high reliability metal oxide resistors, power supply using custom designed toroidal transformer, 2K Basic and 1K monitor in EPROMS and, of course, wire, nuts, bolts, etc.

PSI COMP 80 Memory Expansion System

Expansion up to 32K all inside the computer's own cabinet!
By carefully thought out engineering a mother board with buffers and its own power supply (powered by the computers transformer) enables up to 3.8K RAM or 8K ROM boards to be fitted neatly inside the computer cabinet. Connections to the mother board from the main board expansion socket is made via a ribbon cable.

Mother Board	Fibre glass double sided plated through hole P.C.B. 8.7" x 3.0" set of all components including all brackets, fixing parts and ribbon cable with socket to connect to expansion plug	£39.90
8K Static RAM Board	Fibre glass double sided plated through hole P.C.B. 5.6" x 4.8"	£12.50
	Set of components including IC sockets, plug and socket but excluding RAMs.	£11.20
	2114L RAM (16 required)	£5.00
	Complete set of board, components, 16 RAMS	£89.50
8K ROM Board	Fibre glass double sided plated through hole P.C.B. 5.6" x 4.8"	£12.40
	Set of components including IC sockets, plug and socket but excluding ROMs	£10.70
	2708 ROM (8 required)	£8.00
	Complete set of board, components, 8 ROMs	£78.50

Floppy Disk, PROM programmer and printer interface coming shortly!



PCB size 16.0"x12.5"

Value Added Tax not included in prices

PRICE STABILITY: Order with confidence. Irrespective of any price changes we will honour all prices in this advertisement until May 30th, 1980. If this month's advertisement is mentioned with your order. Errors and VAT rate changes excluded.

EXPORT ORDERS: No VAT. Postage charged at actual cost plus £1 handling and documentation.

U.K. ORDERS: Subsequent to 15% surcharge for VAT. NO charge is made for carriage. Or current rate if changed.

SECURICOR DELIVER: For this optional service (U.K. mainland only) add £2.50 (VAT inclusive) per kit.

UK Carriage FREE

POWERTRAN COMPUTERS

(a division of POWERTRAN ELECTRONICS)

PORTWAY INDUSTRIAL ESTATE
ANDOVER HANTS SP10 3MN

ANDOVER
(0264) 64455

computing today

VOL.2 No.2
APRIL 1980

Editor : Ron Harris B.Sc
Assistant Ed : Henry Budgett
Art Director : Diego Rincon
Production : Dee Camilleri, Loraine Radmore, Lorraine Stout, Paul Edwards, Tony Strakas, Joanne Barseghian.
Advertisement Manager : Chris Surgenor
Advertisement Representative : David Sinfield
Advertisement Production : Sandie Neville
Managing Director : T.J. Connell


```
110  INPUT M
120  PRINT
130  PRINT 'WHAT IS THE INTEREST RATE'
140  INPUT R
180  PRINT
190  DIM N$[3]
200  Z=(B*R)/(E*100)
210  IF (Z+B)/E >= M THEN 230
220  GOTO 280
320  LET T=0
330  LET X=0
340  FOR N=1 TO 37
350  LET I=B*R/1200
360  LET I=INT(100*I+.5)/100
370  LET B=B+I-M
380  LET T=T+I
390  LET X=X+1
400  PRINT X,M,I,M-I,INT(B*100+.5)/100
410  IF B <= M THEN 430
420  NEXT N
430  LET N=N+1
440  LET F=B
450  LET I=B*R/1200
460  LET I=INT(100*I+.5)/100
470  LET T=T+I
480  LET X=X+1
580  PRINT
590  PRINT 'DO YOU HAVE ANOTHER  CASE'
600  INPUT N$
610  PRINT
620  IF N$[1,2]='NO' THEN 640
630  GOTO 30
640  PRINT
```


NEWS 6
If it's worth knowing — it's here

MACHINE CODE 14
Do it a bit simpler

PET BITS 22
. . . . and pieces of useful info.

PROGRAM LIBRARY 26
Order gentlemen please

HP 85 PREVIEW 31
Answer to a dream?

SOFTSPOT SPECIAL 35
Sixteen pages of your listed goodies

PROBLEM PAGE 52
Solve these if you can

PICO–BASIC 55
Smaller and smaller and . . . .

MPUs BY EXPERIMENT 62
Hardlines which work

LETTERS 68
Well, you said it!

LANGUAGE SURVEY 72
How to say what with what and who to!

Distributed by Argus Distribution Ltd. Printed by LSG Limited, Lincoln.

EDITORIAL AND ADVERTISEMENT OFFICE
145 Charing Cross Road, London WC2H 0EE. Telephone 01-437.1002/3/4/5

£39.50* Professional ASCII Keyboards

MODEL KB 756

FULLY ASSEMBLED & TESTED
CASE AVAILABLE

Accessories Available include:—		
Edge Connector	KB15P	£1.95*
Numeric Key Pad	KB710	£7.50*
Plastic Case (Black)	KB701	£12.75*
DC to DC Converter	DC512	£5.00*

* U.K. Orders add 15% VAT on Order total.

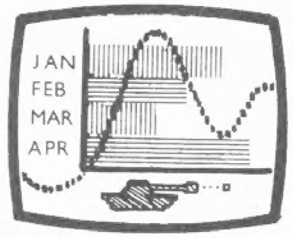
FULL DATA SHEET ON REQUEST

Citadel Products Limited.
Dept. CT. 50 High Street, Edgware,
Middlesex HA8 7EP. Telephone 01-951 1848

CARTER KEYBOARDS

STOP PRESS: Superboard UK101 TRS80 Compatible

COLOUR YOUR NASCOM!



DAZZLING COLOUR GRAPHICS FOR NASCOM 1 & 2

Genuine bit-addressable "pixel" system for straight-forward programming of pictorial or mathematical functions.

8 Colour display plus 8 colour independent background facility. Full documentation with FREE SOFTWARE: powerful sub-routines for vector generation, demonstration program for animated effects. All runs in Nascom 1 without expansion. Complete with UHF Colour Modulator for operation with normal colour TV set. Superior design allows connection to most other microprocessor systems — send us diagrams etc of your b & w video circuitry for free advice. Don't be fooled by the price: this is a top quality product which will transform your computer.

NOW AVAILABLE FOR LIMITED PERIOD AT **£45** PLUS V.A.T.

WILLIAM STUART SYSTEMS Ltd
Dower House, Billericay Road,
Herongate, Brentwood,
Essex CM13 3SD.
Telephone: Brentwood (0277) 810244

BARCLAYCARD VISA

Happy Memories

21LO2	450ns	83p
21LO2	250ns	1.00
2114	450ns	4.25
2114	250ns	4.75
4116	200ns	4.50
4116	150ns	5.50
2708	450ns	4.95
2716	5v	16.95

16K Memory Upgrade Kits for TRS-80, Apple and Sorcerer all at £42.50

Vero S100 Products:	
6 Slot Card Frame with Power Supply	241.36
Prototyping Cards (3 types) each	13.66
Extender Card	21.70
Motherboard with 6 connectors	67.65

IC Sockets:

Pins	Solder	W/W
8	10p	24p
14	11p	36p
16	12p	39p
18	16p	46p
20	17p	58p
22	19p	61p
24	21p	63p
28	27p	70p
40	37p	109p

Ithaca Intersystems Products:	
S100 Z80 CPU Card 4MHz A&T	176.81
S100 Z80 CPU Card 2MHz A&T	150.94
S100 Z80 CPU Card Bare Board	30.19
S100 Video Display Board A&T	125.06
S100 Video Display Board Bare Board	21.56
S100 16 x 2708/2716 A&T	73.31
S100 16 x 2708/2716 Bare Board	21.56
S100 Proto Board (Plated Through)	21.56

See Ithaca's own advertisement for full range

VERBATIM Mini-discs (Pet, TRS-80, etc.) 19.95

SAA5050	14.85
SAA5020	7.15
81LS95	1.50
81LS97	1.54

74LS series:
Lots of them POA

Cannon MIN-D 25 way connectors: Female 1.84
Male 1.60 Hoods 63p. Other sizes available

DIN 41612 Eurconnectors 64 way: Female 3.24
Right-angled Male 1.98. Other Sizes available

DIP Switches: 4, 7 & 8 way all at 85p

We stock a full range of Wire-Wrapping equipment and supplies for you to choose from; we have shelves of books — give us a ring with your requirements and avoid crippling postage charges. Our stocks are too numerous to list here — free price lists sent upon request.

The shop is open from 10 until 6 Monday to Saturday and is worth a visit to catch a surplus bargain — Keyboards at 39.50 today, we may have some left when this gets to print? 18 slot S100 Mother Boards at 22.50?

ALL PRICES INCLUDE VAT. POSTAGE FREE ON ORDERS OVER 10 POUNDS IN VALUE, OTHERWISE ADD 30p. ACCESS AND BARCLAYCARD WELCOME, YOU MAY TELEPHONE WITH YOUR CARD NUMBER AND REQUIREMENTS. TRADE ACCOUNTS ON APPLICATION, GOVERNMENT & EDUCATIONAL ORDERS WELCOME £10 MIN.

DEPT CT
19 Bevois Valley Road, Southampton,
Hants. SO2 0JP Tel: (0703) 39267

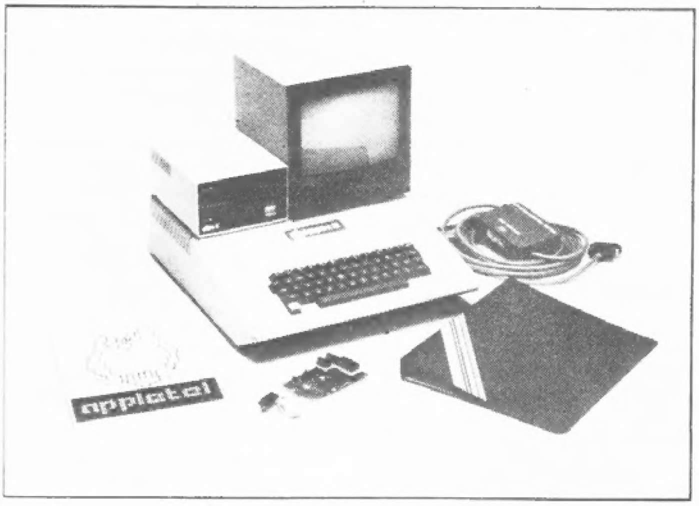
TRS 80 SOFTWARE

The leaders in innovative software

All types of programs for the TRS 80
Model 2 software available soon
Large 27p SAE for our current catalogue

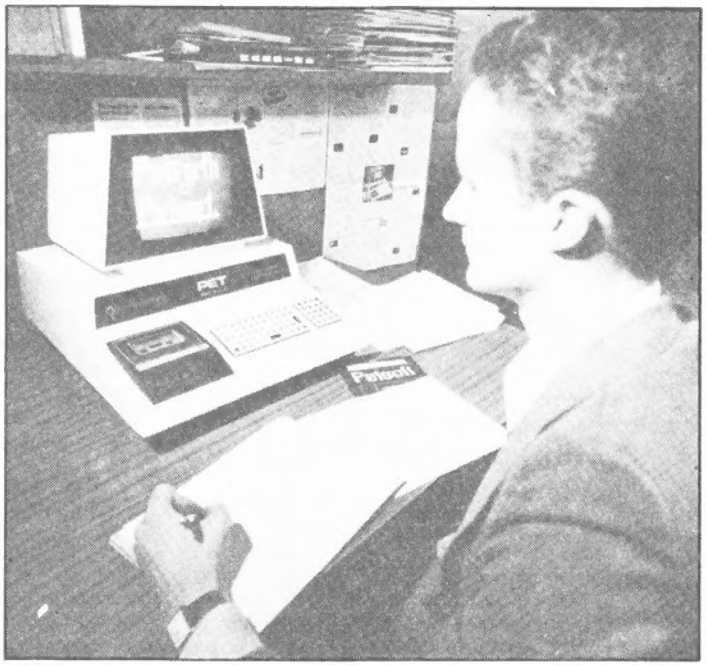
A. J. HARDING (MOLIMERX)

28 Collington Ave.,
Bexhill, E. Sussex.
Tel. (0424) 220391



APPLE CROP

As each month goes by we seem to find more and more bits being made for Apples, a veritable glut of goodies must exist by now. The latest offerings to be added to the mountain are both from Microsense, the main dealers. The first is an ALF Music Synthesiser card which allows you to write your own magnum opus on the screen with a paddle control. You can play about with the pitch, envelope, decay, sustain and volume within the full piano range of eight octaves and then send the completed work to the outside world through your HiFi. Each card can produce three voices and you can have up to three cards. The unit costs £180. The second – and far more important – offering is the new Prestel capability. Owl Computers, in conjunction with the PO and Microsense have modified a standard Apple communications card to handle the Prestel transmissions. The unit has provisional approval and is expected to cost around £600 but this does not include the modem which you will have to rent from the PO. For more details on the Prestel card contacts Mike Gardner of Owl Computers on 0279-52682 or for general Apple info contact Microsense at Finway Road, Hemel Hempstead, Herts.

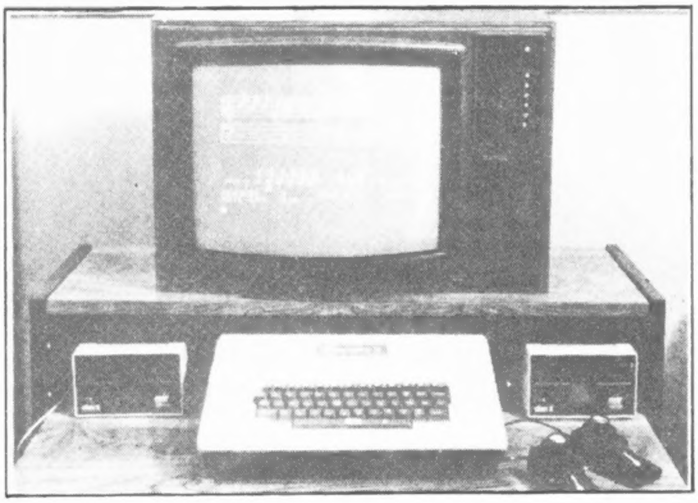


PETSOFT ACT

Cries of "more choice" have now been answered by Petsofts introduction of a FORTH Compiler/Interpreter package for the PET. The implementation consists of a 200 word 'dictionary' where each word is equivalent to a subroutine in BASIC. This allows the user to tailor programs easily and also allows structured programming thus making for more flexible software. The package also includes an assembler and text editor, the cost is a mere £30 + VAT. The parent company, ACT, have further cemented their relationship with the American firm Computhink by jointly producing the new 800 Series machine. At an exceptionally well oiled press reception the august members of the computer press were allowed to play with the system before the dealers had their show. I suspect that this practice will be discontinued owing to the erasing of vital programs on disc. Still the system is very flexible, the 8" floppies give it a very useable one and a bit megabytes and it looks and feels like a GT version of the PET. Perhaps Commodore will think twice about the introduction of their new system into this country? The main benefit that this small business system can offer is its compatability with all the existing software produced by ACT for the PET – a figure of 80% is quoted. Technical details are :– 2 MHz 6502, 46K user RAM, 12" VDU (64 by 30), 122880 addressable graphics points and standard PET graphics, full editing, three programmable fonts, standard parallel printer port, RS232 serial port, disc port for up to four 8" floppies, Extended DOS, Microsoft BASIC with machine code monitor and Tiny Assembler, full ASCII keyboard in upper/lower case and numeric pad. The price for a basic system is quoted at £3,950 with 800K mass storage. For details contact ACT at 5–6 Vicarage Road, Edgbaston, Birmingham B15 3ES.

SHOP TALK

First out of the bag this month is the news that the Byte Shop, who were taken into receivership a month or so back, has been sold off to Comart. Comart are better known for their comprehensive stocks of Cromemco kit among small business users. The chain of existing shops is still trading under the name Byte Shop 1980 and it looks as though the future plans for expansion will go ahead, but maybe not so fast. Second item to crawl out of the mailbag is news that Adda, the West London store, are to open a central London office in Hanover Street. The new address is 1-2 Hanover St., London W1 and the phone is 01-408 1611. The main aim behind the expansion is to offer a better, faster software service to businessmen in the central London area. Midwich, the Nanocomputer people had sprouted a new organisation to handle its small business machines and Apples. The new company is called Siafield and is looked after by Phil Everton at the same address as Midwich, namely 209b High Street, Waltham Cross, Herts EN8 7AY, phone Waltham Cross (97) 29310. Midwich are now dealing solely with the SGS-Ates range of product, including the highly successful Nanocomputer. And last but not least comes news of expansion by Newbear, in the guise of Newbear Books. They have opened a new store in Birmingham, at the Tivoli Centre, Yardley and there is good access and lots of free parking. Shop hours are 9–5 Monday to Friday and you can phone on 021-707 7170 for a booklist or further details.



SINCLAIR HOMES IN

Clive Sinclair, the man who brought you the Mk 14 and the Microvision, among other things, has announced his latest offering to the world. Called the ZX80 it is a Z80A based system with a touch keyboard and built in BASIC. The whole thing is about the size of a small desktop calculator — you can hold it in your hand quite easily. Special items abound because of a radical new design idea, you get single key operation for common BASIC commands, syntax error checking on entry and a full alphanumeric keyboard. However with the price at just under £100 ready built, or £77.95 for the kit, you do have to accept restrictions on the flexibility of the machine. The Z80A is run at 3.25 MHz instead of the usual 4 MHz as this is a convenient multiple of the TV scan frequency, yes the CPU has to look after the TV as well, and the system is not as fast as one would expect. The BASIC has been squashed down from 4K Integer style by using a look-up table system for the command words and by having the character set built in with the monitor. One is not informed how much ROM is used but there is 1K of RAM — equivalent in Sinclair terms to 4K of usual RAM. The design includes the drivers for the VDU, a normal B/W TV but black on white, and a cassette interface for mass storage, at some non-standard rate, but the bus capability is very limited owing to a lack of buffering.

You can add up to 16K of RAM onto the system, it comes in 3K chunks which cost £12 for the board and £16 per K. Because of the architecture it would not be advisable, although possible, to hang exotic peripherals onto the system as the CPU would spend more time looking after them than it would servicing the program. However, at the price it represents a definite opening into computing for schools and colleges — especially in kit form — and as a very high class executive mind stretcher. The kit version is available as from February and the ready built system is promised for March. We have naturally asked for a review sample and will let you know in the near future how the system measures up under test. Sinclair Research can be reached at 6 Kings Parade, Cambridge CB2 1SN and their phone is 0223-312919.

CLUB CALL

I think that our last club survey has produced the highest ever response, we have had over 90% of our forms back now so if you are still holding onto one, send it in. First in the column this month come the additions to the list. The PET education group is run by Dr Chris Smith of the Department of Physiology, Queen Elizabeth College, Campden Hill Road, London W8 7AH. They have about 20 members, membership is free and they have a special interest in CAL, that's Computer Aided Learning according to my dictionary. Changes to the published list are starting to come in as clubs have their AGMs, from the top. . . Southgate Computer Club have a new primary contact, Mr Panos Koumi of 33 Chandos Avenue, Southgate, London N14 7ES. The telephone is 01-882 2983 and they meet on Wednesday and Thursdays fortnightly, membership is £1. The Gwent Amateur Computer Club has got itself a new meeting place, namely 10 Park Place in Newport where they meet each Wednesday night. They now own a couple of systems and as a result the club fees are now £3 or £1.50 for students with a meeting charge of 20p. Their new primary contact is Ian Hazell at 50 Ringwood Hill, Newport, Gwent NPT 9EB and the phone is Newport (0633) 277711 during office hours. SELMIC are now holding regular meetings at the Thames Poly, Churchill House, Greens End, Woolwich on the second and fourth Wednesdays of the month. A door charge of 75p is made for non-members, their primary contact remains unchanged. The North London Hobby Computer Club is moving well, recent talks have included Speech Synthesis and Recognition and Security And Fraud. Future evenings include Artificial Intelligence and Robots on April 9th, CAI on May 7th and the House Computer on June 4th. Meetings are held in the Students Common Room and they start at 7pm. Further details of the club activities can be found in their excellent magazine GIGO. Contact details are unchanged from the survey entry. And whilst on the subject of mags we are now getting Printout, Richard Pawsons new PET extravaganza. Excellent value for money and it can't be too popular with Commodore as he keeps breaking their secrets! Printout costs £9.50 for ten issues and is a far better bet than the official user group who haven't produced a single mag since Richard left to do his own thing — if they have done one I'd love to see it because it hasn't crossed my desk yet.

And finally on the Club scene . . .The Merseyside Microcomputer Group have asked us to point out that they have a vast array of sub-groups as well as the parent body. Full details are given in their regular newsletter but the main ones are Research Machines 380Z (run by Alan Pope), Education Group (run by Mr M Trotter), Apple Special Interest Group, the SC/MP group (run by Bob Perrigo) and the PET special interest group. Contact any of these through the main organisation at the University of Liverpool.

Introducing the PET COLLECTION

... A suite of powerful business programs at a budget price – from ACT Petsoft, the professional software specialists!

1 ACT SALES LEDGER £120

Commodore Disk £95 Cassette version

Full facilities for the maintenance of the Sales Ledger, the preparation of a list of outstanding balances and printing of statements. All data including new customer details, invoices, credits, cash and transfers are entered under step by step guidance on the display screen. Printed results include Audit List, Aged Debtors List, Control Account and Statement.

For 32K PETs

Both Sales Ledger and Purchase Ledger are ... OFFICIALLY APPROVED BY COMMODORE

2 ACT PURCHASE LEDGER £120

Commodore Disk £95 Cassette version

Full facilities for maintenance of the Purchase Ledger, the preparation of a list of outstanding balances and printing of remittance advices. The system produces the following printed results: Audit List, Aged Creditors List, Control Accounts, Purchase Ledger Record, Remittance Advice, Cheques and Payment List. For 32K PETs

PLUS

3 INVOICING WITH STOCK £75

A powerful, easy-to-use system for the CompuThink Disk, handling 1200 or 2400 stock items per diskette.

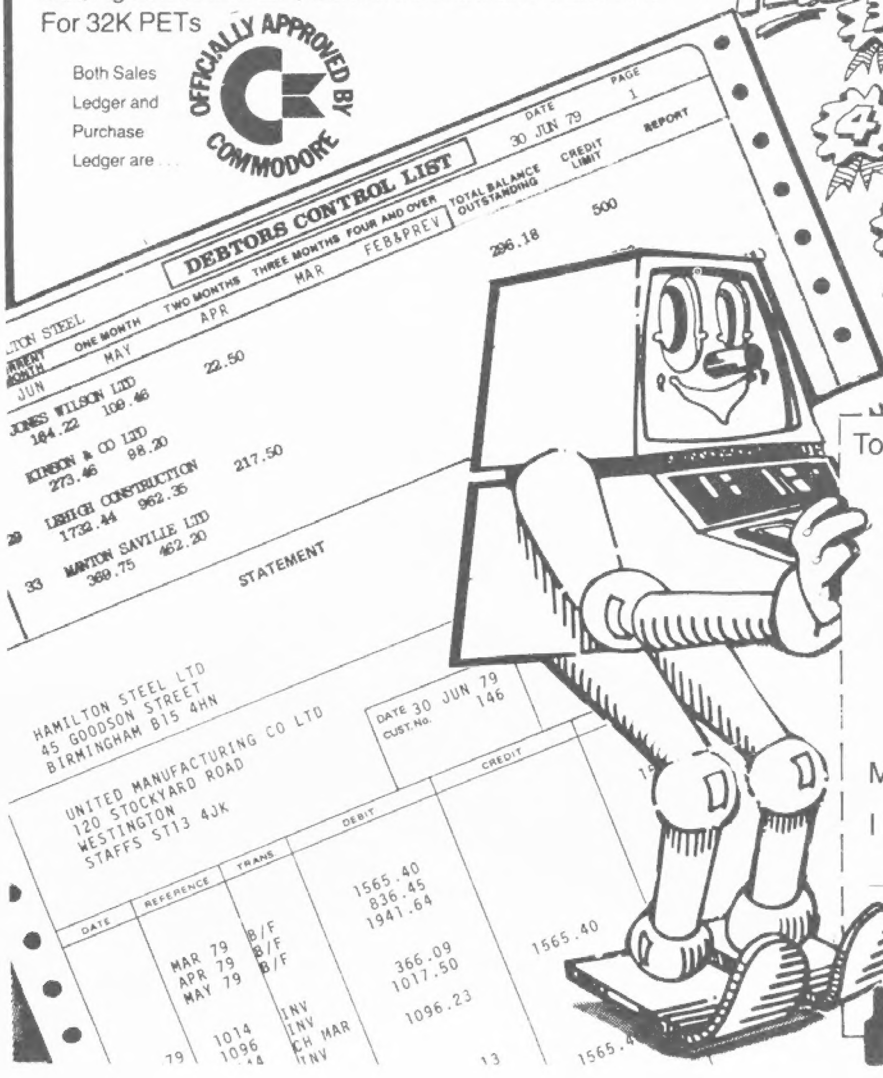
4 PAYROLL 200 £50

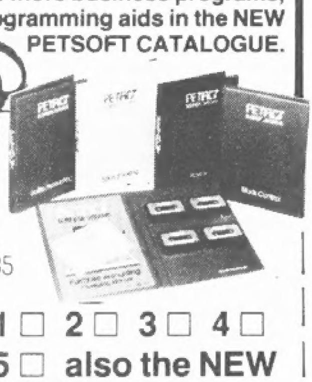
For up to 200 employees, on disk or cassette.

5 WORDCRAFT £325

The ultimate PET Word Processor, now on CompuThink Disk.

... and over 200 more business programs, games and programming aids in the NEW PETSOFT CATALOGUE.





To: ACT PETSOFT
Radclyffe House,
66-68 Hagley Road,
Edgbaston, Birmingham
B16 8PF. Tel: 021-455 8585
Telex: 339396

Please rush me information on ... 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ also the NEW PETSOFT CATALOGUE ☐

My Name is ____________________

I live at ____________________

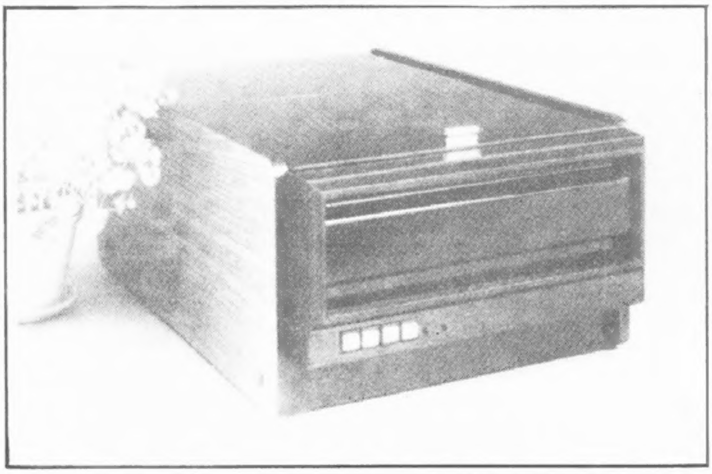
____

Tel. No. ____________________

ACT Petsoft

TAPING IT

Home computer users have now been given more choice in the cassette tape market with the introduction of a new Scotch digital tape from 3M. Designated the 830 it comes in a neat little box and is available in two sizes, C10 and C30. The new style packaging of the tape has also spread to the whole range of 3Ms data recording media and the silver and black design is easily spotted. We have had a sample of the new C30 in the office and it seems to perform excellently, we have not managed to get any LOAD ERRORS on the PET! For details on the whole range of 3M products contact them at 3M House, PO Box 1, Bracknell, Berkshire.



S100 IN A SPIN?

A new disc system for the S100 bus has been announced by Equinox and is claimed to be exceptionally reliable. The unit is the KB10 with 5 MB of fixed and 5 MB of removable storage which is about ten times faster than a floppy. Software currently supportable includes CP/M2, MP/M, FAMOS and OMNIX. The unit can also cope with up to four tape systems along with "unlimited" disc storage, the only limit will be the fact that the unit costs £4950 a go. Details are available from Equinox at Kleeman House, 16 Anning Street, New Inn Yard, London EC2.

OF COURSE

Portsmouth Poly are running a series of courses in the near future. Included in the list are "First Steps" a three day course for engineers on July 2–4, "Second Steps" which is a follow up on 7–9th July, Microprocessor System Design which is a four day course on 17–18th July and a course on Sixteen Bit Micros which runs for two days on 17–18 July. Full details of all these can be obtained from Mrs A P Sizer at the Dept of Electrical and Electronic Engineering, Portsmouth Polytechnic, Anglesea Road, Portsmouth PO1 3DJ. Manchester Poly are also running a seminar on the 28th May in the All Saints Building of the Poly. Called Microcomputing in Research and Higher Education it is aimed at people involved in those areas. Further details are obtainable from Dr G J Boris Allan on 061-228 6171 ext 2457 or direct form him at the poly, Dept of Social Science, Aytoun Street, Manchester M1 3GH. Finally on the topic of conferences COMPSTAT 80 is here. This will be held at the University of Edinburgh between 18 and 22 August and over 300 people have registered so if you want to go you'd better hurry. The main topic of interest is that of computational statistics so if that's your particular scene you can obtain details from the Director, Program Library Unit, University of Edinburgh, 18 Buccleugh Place, Edinburgh EH8 9LN.

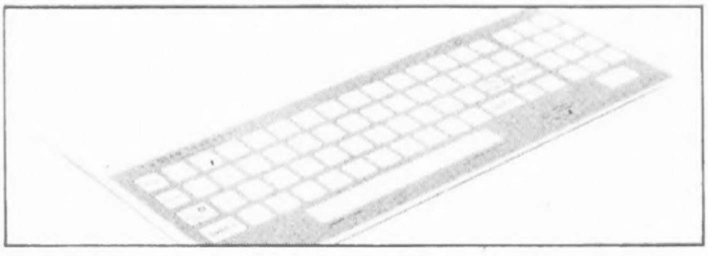
TOUCHY STUFF

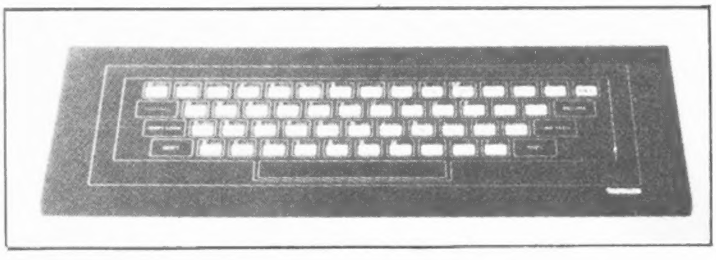
This must be the month for touch sensitive keyboards, two offerings at once, you can't say that we don't bring you the goodies! The first is the Mk 3 touch keyboard from Star Devices that we have on offer in our competition. It is a great improvement on the original Mk 1 that we reviewed back in April last year – not that there was anything wrong with that – and includes some fantastic options. Among the list of possible configurations we find single five volt supply, odd or even parity, repeat key, electronic shift lock, etc and those are just the start. User definable options include RS232 output, fourteen baud rates, on-board ±12 V option for the RS232, TTY character set only, tristate outputs plus many more. Each key is expected to last for at least five million strikes so it's unlikely that you'll wear it out and the touch surface is wipe clean and fully sealed so you can pour coffee all over it. At the bargain price of £48.50 for the basic unit plus your chosen options will you be able to resist it?

The second touch keyboard unit is of American origin and is called TASA. The whole thing is a mere .325 inches in depth and is being stocked by Interface Components of Amersham at a price of £49.50. The unit features a full 128 key ASCII keyboard with electronic shift lock and rollover and the output is supplied in parallel ASCII with strobe to CMOS levels or TT1 with a pull down load for open collector logic. The options are fixed on this unit because it is totally sealed in a lump of plastic but this does mean that it can be used in sterile or hostile environments. Star Devices may be contacted at Unit One, Mill Lane, Newbury, Berks or ring on 0635-40405. Interface components can be reached at Oakfield Corner, Sycamore Road, Amersham Bucks or telephone 02403-5076.

CONFESSION TIME

A couple of slight problems have come to light. In our Feb issue location 0D52 of Malcolm Bell's Logic Emulator should read 0A not A0. The message text for MESS 2 and 3 should also be ignored and re-entered from scratch as corruption occurred. With regard to our Competition the closing date is the end of March so if you haven't sent it in yet get a move on! Several people have also rung up about the number of letters in the 12th clue down, it should be 4 not 5 but most people have worked that out for themselves!





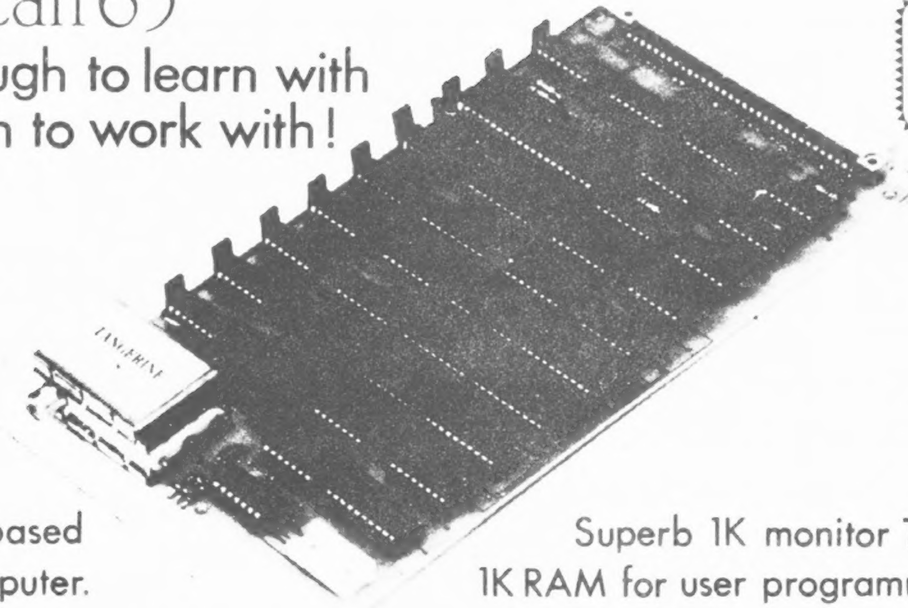
TANGERINE

COMPUTER SYSTEMS LIMITED

microtan 65

Small enough to learn with
Big enough to work with!

£69
plus VAT



A 6502 based microcomputer.
Expands into a system.

Superb 1K monitor TANBUG
1K RAM for user programme, stack and display memory.

VDU alphanumeric display on un-modified domestic TV of 16 by 32 characters.
Optional lower case pack making a total of 128 displayable characters.
Optional chunky graphics pack (64x64 pixels) • Excellent documentation.

MICROTAN 65 outperforms all other small microcomputers in terms of value for money and performance. It is much easier to use as a result of the video display, intelligent keyboard socket, and very powerful monitor. The system grows to become a very useful, complete microcomputer in sensibly priced and very well designed modules. Each module is superbly packaged and with comprehensive hardware and software documentation. The Microtan users manual is A4 size, 136 pages thick, and comes beautifully bound.

TANBUG is probably the most powerful 1K monitor available. Apart from bringing the hardware alive it really does serve the purpose of programme debugging. TANBUG offers memory and register examine, modify and list, block moves, single instruction, multiple non-destructive breakpoints with ability for multiple passes, address offset calculation and many other useful features. MICROTAN 65- More power for your money!

The TANEX expansion board opens the door to a full system as well as including an abundance of features. In its minimum configuration TANEX is supplied with 1K RAM, 16 parallel I/O lines, TTL serial I/O port, cassette interface, 2x16 bit counter timers, memory mapping, data bus buffering, and is fully socketed. When fully expanded TANEX offers all of the above plus a further 6K RAM, 6K ROM, 8K MICROSOFT BASIC, a further 16 parallel I/O lines, 2 counter timers and TTL serial I/O port, and a third serial I/O port with RS232/20mA, full modem control and 16 different baud rates. Check the price list below to see how cheap it is to fully expand the system.

20 Way keypad- plugs into the intelligent keyboard socket and gets you going without the expense of a full ASCII keyboard. Keyboard socket will accept any ASCII keyboard.

Mini-mother board, comes complete with connectors and reset switch.

ORDER FORM:

Qty. Microtan 65 Kit	£79.35
Qty. Microtan 65 Assembled	£90.85
Qty. Tanex (min.con) Kit	£49.45
Qty. Tanex Assembled	£60.95
Qty. Lower case pack	£10.90
Qty. Chunky Graphics pack	£ 7.50
Qty. 20 Way Keypad	£ 8.95
Qty. Mini-mother board	£ 9.95

ALL PRICES INCLUDE VAT. Add £ 1.50 P&P Please.

EXPANSION COMPONENTS:

Qty. Serial I/O Kit	£14.80	6552	£ 9.20
Qty. 2716(5V type)	£28.75	2114	£ 4.80

NAME:

ADDRESS:

BROCHURE AVAILABLE ON REQUEST PLEASE SEND 10p.
TANGERINE COMPUTER SYSTEMS LTD.
FOREHILL, ELY, CAMBS. Tel: (0353) 3633.

TANGERINE

COMPUTER SYSTEMS LIMITED

HELP!

PLEASE DESTROY THIS MAGAZINE

If you do we'll send you a new one (and you'll be helping us!) CT needs some feedback and to get it we must go to you, our readers. In order to go on producing the kind of magazine you want to buy we would like you to tell what you want to see. Every single answer we get is read and considered (honest!) and helps to form the future issues of Computing Today. Why the replacement magazine offer? Well, that is because we know most of you keep your issues in pristine condition and dislike cutting them up — even for a cause as worthy as this. After some head scratching we decided the simplest way to ensure that enough people returned us forms was to provide a new copy in return for the questionnaire. Out and out bribery really.

Please tick below if you wish a new copy and enclose the contents page from this one with your letter.

As a further bribe we are running a draw for 50 of our magnificent T—shirts which you can't enter unless you fill in the form overleaf.

Here is my tick — where's my new CT? ☐

1. Please rate this month's articles on a scale 1(awful) to 9(superb). If you didn't read that particular feature score it 0.

ARTICLE	SCORE	COMMENTS
NEWS		
MACHINE CODE		
PET BITS		
PROGRAM LIBRARY		
HP 85 PREVIEW		
SOFTSPOT SPECIAL		
PROBLEM PAGE		
PICO -BASIC		
MPUs BY EXPERIMENT		
LETTERS		
LANGUAGE SURVEY		

Please asterisk the single best feature in your estimation.

2. How does this issue of CT compare to the last? Please indicate:

MUCH BETTER	SLIGHTLY BETTER	NO DIFFERENT	SLIGHTLY WORSE	MUCH WORSE

3. If you have been a reader since our birth how does this issue compare to our early days?

MUCH BETTER	SLIGHTLY BETTER	NO DIFFERENT	SLIGHTLY WORSE	MUCH WORSE

4. Do you intend to run any of our software from this issue? If yes, which?

1.
2.
3.
4.
5.

5. Which computer system do you own or use at work?

Name . . . . Home use/Work use

6. Which of these topics would you like to see CT cover more in future?

	Need More	About Right Now	Need Less	Should Ignore
Software For Home Systems				
Business Software				
Hardware Designs				
Applications Reports				
Home System Reviews				
Business System Reviews				

Thank you very much for your assistance. Please do not forget to indicate on the previous page if you'd like a replacement issue. In addition if you enter your name and address below we will know where to send your tee shirt if you're one of the 50 drawn.

Name . . . .

Address . . . .

How big is your chest : small, medium or large . . . . (for your tee shirt).

Your Commodore PET System

The Commodore PET is Britain's best selling microcomputer and the most popular choice in every field:–

* In Education for teaching Computer Science and as a teaching aid for other subjects.
* In Science and Engineering for solving problems and for monitoring laboratory equipment.
* In Business the PET system can be put to a wide range of functions including Payroll, Accounting, Statistical Analysis, Stock Control and Word Processing.

Not least of its attractions is the price of a PET – from £550 for a self contained unit, to under £2,500 for the complete system including Floppy Disk Unit and high-speed Printer. Ask your nearest Commodore dealer below for details about Commodore hardware, software and training courses.

Our Dealer* Network

LONDON
Capital Computer Systems, W1. 637 5551
ACE (by Top TV Ltd), SW1. 730 1795
Micro Computer Centre, SW14. 876 6609
Logic Box Ltd, SW1. 222 1122
Sumlock Bondain Ltd, EC1. 250 0505
Da Vinci Computers Ltd, NW4. 202 9630
L & J Computers, NW9. 204 7525
Adda Computers, W1. 408 1611
CSS Business Equipment Ltd, E8. 254 9293
Advanced Management, EC2. 638 9319
Metyclean Ltd, SW1. 828 2511
Microcomputation, Southgate. 882 5104
T.L.C. World Trading Ltd, WC2. 839 3894

HOME COUNTIES
Orchard Electronics Ltd, OXON. 0491 35529
D. L. Chittenden Ltd, CHESHAM. 4441
J. R. Ward Computers Ltd, MILTON KEYNES. 562850
Dataview Ltd, COLCHESTER. 78811
South East Computers Ltd, HASTINGS. 426844
Symtec Systems Ltd, SOUTHAMPTON. 38868
Alphascan Ltd, BANBURY. 75606
Super-vision, SOUTHAMPTON. 774023
Millhouse Designs Ltd, ALTON. (042) 050374
Micro Facilities Ltd, MIDDX. 979 4546
DDM. BRENTWOOD. 230480
Stuart R. Dean Ltd, SOUTHEND. 62707
Alpha Business Systems, HERTFORD. 57423
HSV Microcomputers, BASINGSTOKE. 62444
HSV Microcomputers, SOUTHAMPTON. 22131
RUF Computers (UK), BURGESS HILL. 45211
Wego Computers Ltd, CATERHAM. 49235
T. & V. Johnson, CAMBERLEY. 62506
T. & V. Johnson, OXFORD. 721461
Petalect Electronic Services Ltd, WOKING. 23637/21776
Business Electronics, SOUTHAMPTON. 738248
Amplicon Micro Systems Ltd, BRIGHTON. 562163
Bromwall Data Services Ltd, HATFIELD. 60980/64840
MMS Computer Systems, BEDFORD. 40601
Isher-Woods, LUTON. 416202
Sumlock Bondain, NORWICH. 26259
CSE (Computers), READING. 61492
Oxford Computer Systems WOODSTOCK. 811976

MIDLANDS & STH. HUMBERSIDE
Taylor Wilson Systems Ltd KNOWLE. 6192
Betos (Systems) Ltd NOTTINGHAM 48106
Holbrook Business Systems DERBY. 368088
Lowe Electronics Limited MATLOCK. 2817
Davidson-Richards Ltd. DERBY. 366803/4
Arden Data Processing, LEICESTER. 22255
Tekdata Ltd, STOKE-ON-TRENT. 813631
C.S.M. Computer Systems, BIRMINGHAM. 360 6264
Business & Leisure Microcomputers, KENILWORTH. 512127
Caddis Computer Systems Ltd, HINCKLEY. 613544
Allen Computers, GRIMSBY. 40568
CPS (Data Systems) Ltd, BIRMINGHAM. 707 3866
Camden Electronics, BIRMINGHAM. 773 8240
Cliffstock (Computer Systems) Ltd, WOLVERHAMPTON. 24221

YORKSHIRE & NTH. HUMBERSIDE
Microprocessor Services, HULL. 0482 23146
Microware Computers, HULL. 562107
Computer Workshop, LEEDS. 788466
Hallam Computer Systems Ltd, SHEFFIELD. 663125
Ackroyd Typewriters Ltd, BRADFORD. 31835
Datron Micro Centre, SHEFFIELD. 585490
Yorkshire Electronics Service Ltd, MORLEY. 522181
Sheffield Computer Centre, SHEFFIELD. 53519

NORTH EAST
Dyson Instruments, DURHAM. 66937
Currie & Maughan, GATESHEAD. 774540
Wards Office Supplies, GATESHEAD. 605915
Tripont Associated Systems SUNDERLAND. 73310
Newcastle Computer Services, NEWCASTLE UPON TYNE. (0632) 615325

SOUTH WALES & WEST COUNTRY
Computer and Design, BROADSTONE. 0202 697341
A. C. Systems, EXETER. 71718
Computer Supplies (Swansea) SWANSEA. 290047
Sigma Systems Ltd, CARDIFF. 21515
Devon Computers, PAIGNTON. 526303
Bristol Computer Centre, BRISTOL. 23430
J. A. D. Integrated Services, PLYMOUTH. 62616
Sumlock Tabdown Ltd, BRISTOL. 26685
Radan Computational Ltd, BATH. 318483
T. & V. Johnson Ltd, BRISTOL. 422061

NORTH WEST & NORTH WALES
B & B Computers Ltd, BOLTON. 26644
Megapalm Ltd, CARNFORTH. 3801
Tharstern Ltd, BURNLEY. 38481
Fylde Business Machines Ltd, PRESTON. 731901
Preston Computer Centre, PRESTON. 57684
RPL Microsystems, DOUGLAS. 4247/8

LIVERPOOL
Microdigital, LIVERPOOL. 227 2535
Rockliff Brothers Ltd, LIVERPOOL. 521 5830

MANCHESTER
Cytek (UK) Ltd MANCHESTER. 832 7604
Executive Reprographic Ltd MANCHESTER. 228 1637
Sumlock Manchester Ltd, DEANSGATE. (0618) 834 4233
Computer Workshop, MANCHESTER. 832 2269
Professional Computer Services Ltd, OLDHAM. 061-624 4065
D. Kipping Ltd, SALFORD. 834 6367
Catlands Computers Ltd, Wilmslow 527166

SCOTLAND
Microcentre, EDINBURGH. 225 2022
Thistle Computers, KIRKWALL. 3140
McAllister Business Equipment, EDINBURGH. 336 2402

IRELAND
Softech Ltd, DUBLIN. 784739
Medical and Scientific LISBURN. 77533

*This is a list of dealers participating in associated advertising and not a full list.

commodore

We made small computers big business.

Commodore Information Centre, 360 Euston Road, NW1 3BL. 01-388 5702

M.J. Bell.

Programming in machine code tends to frighten many people away.

This is the first part of a series of articles in response to the demand by novice micro-enthusiasts for advice in programming at a level they can easily understand. Having spent large sums of money on home computers, many are finding it harder to pick up than they had been led to believe by glossy advertisements and, whilst there are many self-teach books available for high level language systems such as BASIC, ALGOL, Pascal and FORTRAN, it is much harder to find instruction in Machine Code programming.

In this first part the Central Processing Unit (CPU) is put under the microscope to gain some insight into how the heart of any micro-system functions. The second part will investigate the 'language of machine code' and ponder on how a mere human can understand its logic. Later parts will be devoted to program writing from first principles, through flowcharting and structuring, to the final documentation stage. This will be illustrated by putting together a program to calculate any monthly calendar from the year 1756 to 9999.

It is hoped that this series will be informative, not only to those who have no alternative to machine code but also to those that have higher order systems that permit user subroutines to be written in code.

The Central Processing Unit (CPU)

The silicon chip, immortalised by political hysteria, is here to stay, and the sooner society is educated to understand its potential and its limitations the better it will be for all concerned. It is no more than a tool in the hands of craftsmen and when properly used will be a tremendous advantage to mankind, but like all complex tools its principle of operation must be understood by the user and from this fundemental will grow experience and innovation.

A CPU is a large scale integrated device (LSI) comprising many thousands of logic gates and 'flip-flop' type memories constructed by advanced techniques onto a single chip of silicon, which is encapsulated in a plastic or ceramic housing. Access to the silicon chip is by way of two rows of 'pins' that are internally connected to the silicon device. Needless to say, in the event of failure repair is impossible.

The CPU's main function is to process data by shifting it in binary form from one set of registers to another set in a manner pre-programmed for each coded operation.

Fig.1 shows the architectural principles of any 8 data-bit wide CPU with a 16 bit address capability. (A 16 data-bit wide CPU is very similar with the data bus expanded to twice the size. The principle of operation is identical.

There are 3 main information paths which are used for control and data transfer to external hardware.

1. CPU and system CONTROL signals
2. 16 bit address bus
3. 8 bit data bus.

All these leads are connected to external hardware that is responsible for the control and storage within the system.

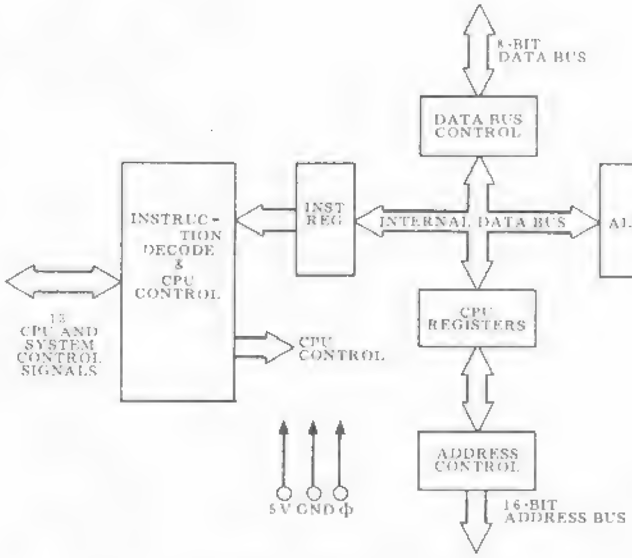


Fig.1. A typical 8 bit CPU's architecture. Each of the main areas is discussed in the text.

One of the important design factors to be remembered if designing or expanding a system is that these leads have a very low power handling capability and great care must be taken to 'buffer' all interconnections.

CPU And System Control Signals

This group of interface connections is used to input such system controls as the 'clock', the Read/Write logic, Single Step logic, Dynamic Refresh etc and the supply voltages. It is not within the scope of this series to pay too much attention to this group as they are more relevant to system design, but further attention will be paid to the 'clock' and the 'dynamic refresh' later on this part.

The Address Bus

The address bus has 16 three-state (tristate) outputs which can be wired to external memory devices within the system. Output from the CPU on these leads is the binary address of the memory location that is to be interrogated either for the purpose of reading data from, or writing data to, that part of the memory. The maximum binary output from the 16 leads is 1111111111111111 or FFFF for short. In decimal terms this number is 65,536, and so this is the maximum number of memory locations possible. In computing terms this is confused still further by calling it 64K memory locations. This apparent anomaly is caused by the internal construction of memory chips which conforms to a matrix format suitable for binary decoding. For instance, a 1K memory contains 1024 cells which are arranged as 32 rows of 32 cells. Not all of these 64K address locations need be used for user program storage, in fact most systems have large areas reserved for the

system monitor and other 'firmware' such as BASIC interpreters. A particular address may even be allocated to control some external machinery, like a modem or random number generator, or additionally it might be the address used for an Input/Output port.

Data Bus
The data bus has 8 tristate connections which can serve as both outputs or inputs as the CPU dictates. These leads transfer data to and from memory devices under the control of the system clock. The maximum size of the binary data ranges from 00000000 to 11111111 which is FF or in decimal terms 256. Any number of memory devices and/or external ports can be connected to the data bus provided it has been adequately buffered as described earlier.

How It Functions

All digital computers work in an orderly manner, and to ensure that data is manipulated only when the sending and receiving hardware is ready a very stable clock pulse is required. It is possible to work a CPU in a 'step-by-step' mode but this is usually confined to the single stepping of instructions rather than clock pulses. A clock pulse can be derived internally in the CPU as with the ROCKWELL 6502, or externally as with the Z80 & 8080. In both cases a crystal at about 16 MHz is used and this is divided down to the normal operating speed of 1, 2 or 4 MHz for most systems.
The CPU uses the clock pulses in the manner shown below:–

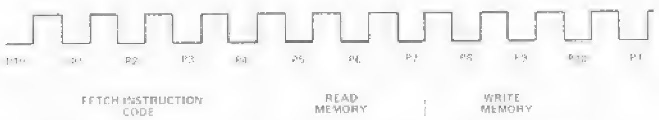


Fig.2. How your CPU gets clocked! This is the vital heart of any system and care must be taken.

The typical timing chart shown in Fig.2 has 10 clock pulses to a complete instruction cycle. The first four pulses are used to decide which operation is required to be carried out, for example a 'shift data', 'arithmetic operation' etc. This is decided in the Instruction Decode & CPU Control area and the Instruction Register shown in Fig.1. During the next three clock pulses data is moved from the specified memory location into the CPU, acted upon, and the final result is written back into memory during the last three clock pulses.

It is not generally necessary for the programmer to have any more detailed knowledge of the CPU timing sequence, but it is essential that the total number of cycles (known as Machine Cycles) is known for each instruction. This information is given in the various manufactures applications documentation for each CPU. Together with the known system operation speed it is possible to calculate the execution time of each instruction.

eg INSTRUCTION; Move the contents of register A to register B
No. of clock pulses; 4
SPEED of SYSTEM OPERATION; 2 MHz ie 0.5 uS per clock pulse
TIME TAKEN; 4 x 0.5 uS = 2 uS

For most home programs the length of time taken to run a routine is not important, but where a timing circuit like a clock or delay element is incorporated the instruction timings become a critical part of the program design. Another application of this is interworking with peripheral equipment such as modems (modulators/demodulators), printers and floppy disc units.

The CPU Internal Registers

The number and form of internal CPU registers vary from one device to another, but typical configurations are shown using the Zilog Z80 and the Motorola 6800 CPU devices, as examples.

Main Registers		Alternative Registers	
Accumulator	Flags	Accumulator	Flags
A	F	A'	F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'

general purpose registers

Interrupt Vector I	Memory Refresh R
INDEX REGISTER	IX
INDEX REGISTER	IY
STACK POINTER	SP
PROGRAM COUNTER	PC

special purpose registers

Fig.3. The Z80's array of registers. You don't often need them all.

For those users of the 8080 device they can consider just the MAIN REGISTERS and the SPECIAL PURPOSE REGISTERS of the Z80 as the alternative set and the I and R registers are not provided. The Z80 registers B, C, D, E, H and L are each 8 bit wide and can be used for data storage. Additionally they can be combined into REGISTER PAIRS as BC, DE and HL for storage of 16 bit wide data such as 16 bit memory addresses or arithmetic arguments. The INDEX REGISTERS, PROGRAM COUNTERS and STACK POINTERS are also 16 bit wide and are used for holding memory addresses. The use of INDEX registers will be covered fully in Part 2 of this series. The PROGRAM COUNTER holds the address of the current instruction that is being executed and is either incremented at the end of the machine cycle, or updated as the program dictates.

The Stack Pointer is used to hold the address of the bottom of a Last In First Out (LIFO) file which is situated in a reserved area of the user memory. This file is required by the CPU to keep track of the return addresses when subroutine calls are made, and can be used to great advantage for the temporary storage of data using the PUSH and POP instructions. Again, this will also be explained in detail when the machine code instructions are investigated in Part 2. The configuration of the Motorola 6800 shown in Fig.4 gives some idea of how much different designs vary. The Program Counter, Stack Pointer and Index Registers are much the same but there are no general purpose registers provided internally. It is necessary to use addresses within the user memory area for any short term storage of data. The 6800 is, however, provided with dual Accumulators compared to the single one on the Z80, and this makes up for some of the deficiency of registers. The Accumulator is the most important of all the internal registers and will be looked at in more detail.

The Accumulator

The accumulator, or 'register A' as it is in known in Z80 and 8080 jargon, is the most used of all the 8 bit registers. It forms the base for all instructions except those that transfer

data between other registers. For example data to be output or input from a port is frequently stored temporarily in the accumulator. It is also the register used for arithmetic or logic operations, and in all cases the result of one of these operations is returned to the accumulator for the next operation. It will be seen when we discuss the system monitors in Part 3 that it is usual to store in this register any data that will primarily be acted upon in a monitor subroutine such as Delay, String, Input or Output.

The Flag Registers

The Flag register or Condition Code Register as it is known in the 6800 world is different from all other registers in that it cannot be written into by a user. It is used to indicate certain conditions that may have occurred during the execution of the previous instruction, like the result being zero or less than zero. It can indicate if the result of a comparison is true or not true and furthermore it can indicate vital information required for arithmetic operations such as half carry and overflow. Acting on the information from these flags a program can make decisions as to which subsequent program path it is to follow, thus giving great flexibility to program writing.

A list of the functions of these flags is given below, although it must be remembered that some may not be included in your system.

CARRY — This flag is set if, as a result of an add, subtract or compare instruction the result causes the accumulator value to pass through zero either from 1 down to FF or from FF up to 1. This flag is also used as a temporary store in the shift and rotate instructions.

ZERO — As its name suggests this flag is set *only* when the result of an instruction is zero.

SIGN — In order to determine if a number is negative or positive the CPU looks at the most significant bit of the accumulator. If it is positive the most significant bit is '0', if negative the bit is '1'. The sign bit is a reflection of the most significant accumulator bit and therefore indicates the sign of a number.

PARITY/OVERFLOW — Parity checking of numbers of inputs is frequently carried out if there is any possibility of corruption, if for example the input is from magnetic tape where 'drop-out' can corrupt data. The parity/overflow bit is used to indicate if the result of an arithmetic or logical operation is odd or even. This flag is also used to indicate an overflow as a result of 'two's compliment arithmetic' operation.

HALF CARRY & ADD/SUB — These two flags are of academic interest only as they cannot be used by a programmer. The CPU interrogates them when carrying out a decimal adjust operation. This will be described in Part 2.

Interrupt Register

Before looking at the interrupt register a few words on what an interrupt is. If a system is being used to control a number of external circuits it is possible that at any time one or more of those circuits will require the system to service it. When this is detected by the system it is required to jump to the correct part of the program to deal with that circuit. Its normal operation is interrupted. The interrupt register is used to carry the last 8 bits of an interrupt address vector. The system port will carry the other 8 bits thereby specifying a memory address in the program.

Memory Refresh Register (Z80)

A very useful register provided more for the system designer than the programmer. External memory has often been mentioned wihtout explanation but there it must be considered more fully. As has been seen a CPU has a very limited capacity of memory that can be accessed by the program. It is therefore a prime requirement that a large external field of memory is available to store both data and the precise order of program instructions. This memory field can comprise of two types of memory; Read Only Memory and Random Access Memory (or Read/Write memory). The first is preprogrammed and contains things like system monitors or high level language programs, whilst the second can be used by the programmer for data. Of this second type, usually referred to as RAM there are two types; the static RAM and the dynamic RAM.

Without going into too much detail because memory devices are a subject in their own right, a static RAM will hold its data secure as long as the supply voltage is maintained. A dynamic RAM will lose its data very rapidly unless it is constantly *refreshed* by having its data continually updated. It is to facilitate this constant memory refreshing that the Z80 provides the Memory Refresh Register. Where it is not provided on other systems a separate circuit element must be used if dynamic memory is required.

Arithmetic And Logic Unit (ALU)

This is the last of our building blocks to examine and is the one that handles the arithmetic and logic capability of a CPU. It is a serial device and works very much like a pocket calculator taking in data from a store (accumulator) carrying out the function, and then outputting the answer. In this case back into the accumulator. ALU's to date perform only the most simple functions of arithmetic but newer devices on the market are beginning to include multiplication and division, and there is little doubt that in the future this is one area of development that will see a great deal of change.

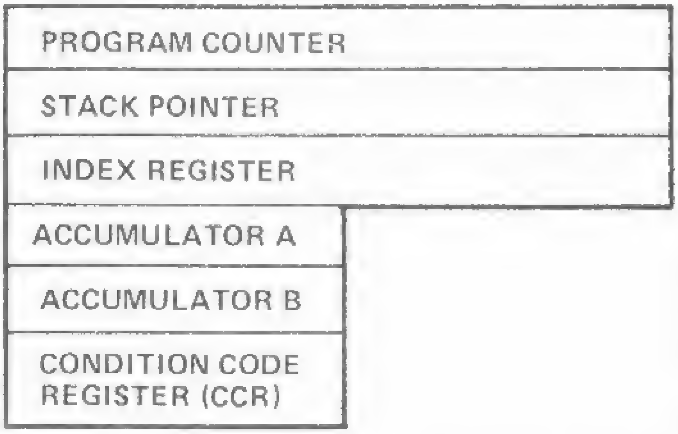


Fig.4. The 6800's selection, less than the Z80 but still enough for most purposes.

BIT	7	6	5	4	3	2	1	0
	S	Z	X	H	X	P/V	N	C

WHERE S = SIGN FLAG
H = HALF CARRY FLAG
N = ADD/SUBTRACT FLAG
Z = ZERO FLAG
P/V = PARITY/OVERFLOW FLAG
X = UNUSED BITS

Fig.5. The status word exposed. Understanding of its function is vital.

computing today

WHAT TO LOOK FOR IN THE MAY ISSUE ON SALE APRIL 11TH.

STOCKMARKET SIMULATION:

If you lie awake at night dreaming of what you would do with that £1000 gift that will always arrive in the post tomorrow, dream no longer. In a remarkable game program written for the TI59 we have a complete stockmarket where you can buy and sell, wheel and deal, bear and bull to your hearts content.

Whilst the stock exchanges of reality are curious and wonderful establishments this game is based on proper theories and a full and detailed explanation of the processes needed is given. So, if you don't own a TI59 do not despair you will be able to use the information to implement the game on virtually any programmable system.

So if you want to be among the market leaders next month invest your sixty pence in our May issue, it could be the best investment you'll ever make.

DRIVEN TO DESPAIR?

Does your car or motorcycle seem to want more money than you bargained for? The author of our Home Finance program presents a second offering which will cater for your automotive expenses.

The program runs on the family PET but is easily adaptable to any BASIC using system with the PETs facilities. Access is available to a number of accounts for details of repair and servicing costs and reminders are given about the life expectancy of wearable items such as tyres.

If you depend on your car and can't account for the money you spend, load up and discover where its all going to. Rumour has it that Panther De Ville owners with that optional PET may be buying all copies so get to the newsagents early.

TRITON REVISITED

ETIs own computer system is over a year old now, and changes have been made since its conception that make it rather more than a single board computer.

In our continuing series of owners reports on popular machines John Hiscott takes his system through the stages of development and lays his observations open to the public eye.

TERMINOLOGY

No, that's not the art of making connections, but a glossary of the "hundred most used terms" in home and hobby computing. Many of our enquiries start out with, 'I can't tell the difference between RAM and ROM' so we decided to reveal all.

As an aid to simulating conversation this pull out extra should not be missed, you might even learn the elusive art of confusion!

A collection of ideas from PET owners.

FIRST BIT

R. Cason

As any PET user knows the most annoying part of loading from Cassette is waiting for the FOUND'— — — — — — —' message. This is mainly due to the lack of a tape counter, but even with a counter you would not know if the PET had passed the Program Header. You can sometimes miss the header — wait several minutes — only to find you are on blank tape.

A Solution

My method is as follows:— Connect a Soundbox to the user Port Pin 6 (Cassette No. 1 Read) The Soundbox connection is Pin M (CB2 Line)

On both SAVE and LOAD you can then hear the following:

a) The Header Tone
b) The Header Token
c) The Header 'Title'
d) The Program DATA
e) The 'Half Way Point'
f) Second copy of DATA
g) The end of file Token

By using the F. FWD, PLAY and REW keys you can then locate the header on a multi-program tape — Press Play — and wait. If you do not get the message FOUND '— — — — — —' at the Header Title stage, rewind slightly and try again. Using this method you can CUE the tape to the right position.

Other advantages are that you can also hear:

a) DROPOUTS
b) CROSSTALK
c) NOISE
d) VARIATION in PITCH due to tight Cassettes.
e) The difference between DATA and PROGRAM tapes.

This is an invaluable aid, and is best implemented by fitting a small toggle switch to the cover of the user port connector.

i.e.	Position 1	SOUND (Pin M)
	Position 2	OFF (No Connection)
	Position 3	CASSETTE (Pin 6)

With Pin N being the 'earth'.

Spare Tape

For those who like to keep a 'Working Copy' of their programs in addition to the 'MASTER' a separate cassette is an advantage. I use an Hitachi TRQ 299 which has an automatic level control (ALC) and a Cue and Review facility. In my case the ALC gives perfect results on the PET recordings every time. The Cue and Review facility allows you to fast wind using Cue to find the 'nth' program on the tape.

Position the header using Review and transfer the tape to your PET Cassette. Perhaps somebody will devise a method to convert the PET Cassette to 'Cue and Review'.

Incidentally can anybody suggest a method of recovering data from a Program tape, on which the header and part of the first copy of DATA has been erased? (Caused by pushing RECORD instead of PLAY).

SECOND BIT

Jim Cocallis

The following routine allows a user to display PET memory from address 0000 to address 65536. It has been written in machine code for speed reasons: BASIC is adequate but it is rather slow.

The routine used is a good example of simple machine code programming and because I am sure many of you are playing with the idea of starting to investigate machine code I will itemise the program step by step. First the whole routines :—

Location	Mnemonic	Location	Mnemonic
033A	LDA #00	0366	LDX #01
033C	STA $DA	0368	LDY #00
033E	STA $DB	036A	LDA (DA),Y
0340	LDA #78	036C	STA (DC),Y
0342	STA $DC	036E	INY
0344	LDA #0E	036F	CPY #00
0346	STA $E84C	0371	BNE $036A
0349	LDA #93	0373	CPX #00
034B	JSR $FFD2	0375	BEQ $037E
034E	LDA #80	0377	DEX
0350	STA $DD	0378	INC $DB
0352	LDX #00	037A	INC $DD
0354	LDA $038A,X	037C	BNE $0368
0357	JSR $FFD2	037E	JSR $FFE4
035A	INX	0381	BEQ $037E
035B	CPX #0F	0383	INC $DB
035D	BNE $0354	0385	CMP #20
035F	LDX $DA	0387	BNE $0349
0361	LDA $DB	0389	RTS
0363	JSR $DC9F		

038A to 0399 = Symbol Table

$ = Hex. address   # = Hex. numbers

Explained Away

Frightening isn't it? Let's make it understandable by giving it some meaning. The first column (headed location) shows the address at which the first part of the next column's content is held in memory e.g. 033A holds the 8 bit code representing the mnemonic LDA 033B holds the 8 bit code for 00 and so on. The second column shows the mnemonics used by the assembler programmer to assemble his program. The mnemonic is used as an easy way to recall the binary code which the micro understands, viz LDA is A9 in hex and 10101001 in binary. It can be seen that it would be very difficult to remember the binary code, a little less difficult to recall the hex code and considerably easier to recollect LDA. The disadvantage is that a special program is needed to convert the mnemonics into the binary code.

Now that the layout is clear we can get on with the hard part; devising and coding a program. I needed to see how BASIC stored a program and naturally I wrote a program using that language to display the contents of RAM. A problem arose; Microsoft BASIC in PET is PEEK protected and I was not able to look at the way in which it is stored. A secondary problem was the time taken to print 1000 bytes onto the screen, BASIC tends to be slow if it is PEEKing a location and then printing the contents of the variable onto the screen. The routine devised was not good enough to satisfy my need so I decided to look for a quicker way and eventually the only good solution was to write a program using machine code (MC). Before I could do so I needed to know what routines were available to me using the ROMs in the PET. It is not much use writing a small program to output results to the screen if a routine is already

available. There are numerous sources of information available, IPUG, PET User Groups, books and magazines etc., and after consulting many of them I was able to sit down and write the first version. It did not work!! After some corrections were made it worked and it is this final version which is shown in this article.

Using Your ROM

The main ROM routines used are as follows:

$FFD2 This routine prints out the contents of the Accumulator.

$FFE4 Get a character and place it in the Accumulator, if no character then place 0 in the Accumulator.

$DC94 Take the contents of the X register plus the contents of the Accumulator and convert them into a decimal number then print the number onto the screen.

Locations $DA to $DD are not used by BASIC and can safely be used for MC programs to store variables or constants.

To business; (I will use the line address as a reference)

033A Loads the Accumulator (LDA) with the hex. number 00

033C and 033E Store the contents of the Accumulator in addresses $DA and $DB. This is the start address 0000.

0340 LDA with the low order byte of the screen location which will hold the first character to be output.

0342 STA in a location which can be accessed later.

0344 and 0346 Set mode to lower case graphics: hex 0E = 14 and hex E84C = 59468.

0349 LDA with a hex code representing a character.

034B Jump to a subroutine which converts the contents of the Accumulator and prints it on the screen – 93 is the code for 'Clear screen'.

034E LDA with the high order byte of the screen start address e.g. hex 8078 = 32888.

0350 STA high order byte for later use. This routine prints "Starting byte = "onto the screen. It does so by adding the value of the X register to

0352 to 035D the address specified ($038A) and fetching the contents of that address which is printed on the screen. The counter X is incremented by one and tested for equality to 15. If it does not equal 15 then the next character is fetched. Check the number of characters in the output above.

035F and 0363 Takes the value held in locations $DA and $DB and prints it after completing the routine above.

0366 LDX with the count value (do it twice).

0368 LDY with 00 (count the spaces).

036A This instruction takes the value held in Y and adds it to the address held in $DA plus the next address $DB viz $DA contains 00, $DB contains 00. The address held in these two locations is 0000 and Y has the value 00. The address to be accessed is therefore 0000 + 00 = 0000. If Y held 19 then the figures would be 0000 + 19 = 0019. This method allows us to access 256 locations before resetting.

036C This routine stores using the same indirect instruction used above. It stores 256 bytes starting at $807F (prints onto screen).

036E Increase the counter by one.

036F This, comparison makes use of the fact that incrementing an 8 bit register which contains 11111111 causes it to reset to 00000000. The test for zero ensures that a full 256 cycles is done.

0371 If the Y register is not equal to 00 then go back and do the whole thing again.

0373 This line compares the X register to zero.

0375 If X = 0 then branch to another routine.

0377 If X = 0 then Decrement X and thereby reduce it to zero. The effect of this is to ensure that the whole routine is only completed twice and only displays 512 bytes of memory. Try changing the value of X in $0366 and see what happens.

0378 If you refer back to the explanation of lien 036A you will see that the instruction uses the next location to the one shown, $DA uses $DB. We must increment the values in the low order byte location if we want to progress through memory. This line does this once for every 256 loops.

037A This line increments the screen address as above

037C If Accumulator is greater than zero do it again

037E Get a character

0381 If no key pressed go back and test again

0383 Increment the value in $DB (high byte start address)

0385 Was the key pressed a 'space'

0387 If not a space then start routine again, displays the next 512 bytes.

0389 Return to BASIC

038A to 0399 These addresses contain the codes of the letters needed to print out the message.

That's the whole thing and I am sure that you will understand it a lot better the next time you read this article. I show below the hex dump relating to this routine. To convert it for input via BASIC convert all the codes into decimal form and poke each one into successive locations using a For-Next loop.

Note

One last thing, this routine is not the best possible routine and I am sure that it could be improved upon, however, it does the job it was intended to. If you do improve it please let me know.

033A	A9	00	85	DA	85	DA	A9	78
0342	85	DC	A9	0E	8D	4C	E8	A9
034A	93	20	D2	FF	A9	80	85	DD
0352	A2	00	BD	8A	03	20	D2	FF
035A	E8	E0	0F	D0	F5	A6	DA	A5
0362	DB	20	9F	DC	A2	01	A0	00
036A	B1	DA	91	DC	C8	C0	00	D0
0372	F7	E0	00	F0	07	CA	E6	DB
037A	E6	DD	D0	EA	20	E4	FF	F0
0382	FB	E6	DB	C9	20	D0	C0	60
038A	53	54	41	52	54	49	4E	47
0392	20	42	59	54	45	20	3D	20

computing today

technical book service

What Is A Microprocessor?
2 Cassette tapes plus a 72 page book deal with many aspects of microprocessors including Binary and Hexadecimal counting, Programming, etc. **£12.00**

Adams, C. BEGINNERS GUIDE TO COMPUTERS AND MICROPROCESSORS WITH PROJECTS £6.05
Understanding building programming and operating your own microcomputer.

Ahl, BASIC COMPUTER GAMES £5.40

Albrecht, B. BASIC FOR HOME COMPUTERS. A self teaching guide £5.45
Shows you how to read, write and understand basic programming language used in the new personal size microcomputers.

Albrecht B. BASIC. A self teaching guide (2nd edition) £5.45
Teach yourself the programming language BASIC. You will learn how to use the computer as a tool in home or office and you will need no special maths or science background.

Alcock, D. ILLUSTRATING BASIC £3.00
This book presents a popular and widely available language called BASIC, and explains how to write simple programs.

Altman, I. MICROPROCESSORS £10.80
Gives a general overview of the technology design ideas and explains practical applications.

Altman, L. APPLYING MICROPROCESSORS £15.50
Follow volume which takes you into the second and third generation devices.

Aspinall, D. INTRO TO MICROPROCESSORS £6.55
Explains the characteristics of the component.

Barden, W. Z-80 MICROCOMPUTER HANDBOOK £7.75

Barden, W. HOW TO BUY AND USE MINICOMPUTERS AND MICROCOMPUTERS £7.90
Discusses these smaller computers and shows how they can be used in a variety of practical and recreational tasks in the home or business.

Barden, W. HOW TO PROGRAM MICROCOMPUTERS £7.25
This book explains assembly language programming of microcomputers based on the Intel 8080, Motorola MC6800 and MOS Technology MCS6502 microprocessor.

Barna, A. INTRODUCTION TO MICROCOMPUTERS AND MICROPROCESSORS £8.60
Provides the basic knowledge required to understand microprocessor systems. Presents a fundamental discussion of many topics in both hardware and software.

Bibbero, R. J. MICROPROCESSORS IN INSTRUMENTS AND CONTROL £12.60
Introduces the background elements, paying particular regard to the dynamics and computational instrumentation required to accomplish real-time data processing tasks.

Lancaster, D. TV TYPEWRITER COOKBOOK £7.75
An in-depth coverage of tv typewriters (tv's) the only truly low cost microcomputer and small display interface.

Lancaster, D. CHEAP VIDEO COOKBOOK £6.50

Lesea, A. MICROPROCESSOR INTERFACING TECHNIQUES £8.70

Leventhal. INTRO TO MICROPROCESSORS £17.00

Lewis, T. G. MIND APPLIANCE HOME COMPUTER APPLICATIONS £4.90

Libes, S. SMALL COMPUTER SYSTEMS HANDBOOK £5.90
The Primer written for those new to the field of personal home computers.

Lippiatt. ARCHITECTURE OF SMALL COMPUTER SYSTEMS £4.50

Moody, R. FIRST BOOK OF MICROCOMPUTERS £3.85
(the home computer owners best friend).

McGlynn, D. R. MICROPROCESSORS – Technology, Architecture & Applications £9.20
This introduction to the computer-on-a-chip provides a clear explanation of the important new device.

McMurran, PROGRAMMING MICROPROCESSORS £5.50
A practical programming guide that includes architecture, arithmetic/logic operations, fixed and floating point computations, data exchange with peripheral devices' computers and other programming aids.

Monro, INTERACTIVE COMPUTING WITH BASIC £3.65

Nagin, P. BASIC WITH STYLE £4.15
Programming Proverbs. Principles of good programming with numerous examples to improve programming style and producing.

Ogdin SOFTWARE DESIGN FOR MICROCOMPUTERS £7.20

Ogdin. MICROCOMPUTER DESIGN £7.25

Peatman, MICROCOMPUTER BASE DESIGN £5.70

Peatman, J. B. MICROCOMPUTER BASED DESIGN £21.00
This book is intended for undergraduate courses on microprocessors.

Peckham, HANDS ON BASIC WITH A PET £9.00

Peckham, BASIC – A HANDS ON METHOD £6.95

Bursky, D. MICROCOMPUTER BOARD DATA MANUAL £6.00

Coan, J. S. BASIC BASIC £7.50
An introduction to computer programming in BASIC language.

Coan, J. S. ADVANCED BASIC £7.30
Applications and problems.

Ditlea, A SIMPLE GUIDE TO HOME COMPUTERS £4.10

Freiberger, S. CONSUMERS GUIDE TO PERSONAL COMPUTING AND MICROCOMPUTERS £5.75

Frenzel, L. GETTING ACQUAINTED WITH MICROPROCESSORS £7.25
This is an invaluable book for those who want to know more about hobby and personal computing.

Gilmore, C. M. BEGINNERS GUIDE TO MICROPROCESSORS £4.90

Grossworth, BEGINNERS GUIDE TO HOME COMPUTERS £4.50

Gosling, R. E. BEGINNING BASIC £3.45
Introduces BASIC to first time users.

Graham, N. MICROPROCESSOR PROGRAMMING FOR COMPUTER HOBBYISTS £7.15

Hartley, INTRODUCTION TO BASIC £2.80

Heiserman, D. L. MINIPROCESSORS FROM CALCULATORS TO COMPUTERS £4.95

Hilburn, J. L. MICROCOMPUTERS, MICROPROCESSORS, HARDWARE, SOFTWARE AND APPLICATIONS £17.40
Complete and practical introduction to the design, programming operation, uses and maintenance of modern microprocessors, their integrated circuits and other components.

Klingman, E. MICROPROCESSOR SYSTEMS DESIGN £16.95
Outstanding for its information on real microprocessors, this text is both an introduction and a detailed information source treating over a dozen processors, including new third generation devices. No prior knowledge of microprocessors or microelectronics is required for the reader.

Kemeny, J. G. BASIC PROGRAMMING £6.70
A basic text.

Korn, G. A. MICROPROCESSOR AND SMALL DIGITAL COMPUTER SYSTEMS FOR ENGINEERS AND SCIENTISTS £19.45

This book covers the types, languages, design software and applications of microprocessors.

Duncan. MICROPROCESSOR PROGRAMMING AND SOFTWARE DEVELOPMENT £14.15

Rao, G. U. MICROPROCESSOR AND MICROPROCESSOR SYSTEMS £20.75
A completely up-to-date report on the state-of-the-art of microprocessors and microcomputers written by one of the leading experts.

Rony, P.H. THE 8080A BUGBOOK: Microcomputer Interfacing & Programming £8.35
The principles, concepts and applications of an 8-bit microcomputer based on the 8080 microprocessor IU chip. The emphasis is on a computer as a controller.

Scelbi. 6800 SOFTWARE GOURMET GUIDE AND COOKBOOK £8.90

Scelbi. 8080 SOFTWARE GOURMET GUIDE AND COOKBOOK £8.90

Scelbi. UNDERSTANDING MICROCOMPUTERS £8.75
Gives the fundamental concepts of virtually all microcomputers.

Spencer, GAME PLAYING WITH BASIC £4.85

Schoman, K. THE BASIC WORKBOOK £3.85
Creative techniques for beginning programmers.

Sirion, D. BASIC FROM THE GROUND UP £6.20

Soucek, B. MICROPROCESSORS AND MICROCOMPUTERS £19.40
Here is a description of the applications programming and interfacing techniques common to all microprocessors.

Spracklen, D. SARGON £9.90
A computer chess program in Z-80 assembly language.

Titus, MICROCOMPUTER ANALOGUE CONVERTER £7.60

Titus, 8080/8085 SOFTWARE DESIGN £7.60

Tracton. 57 PRACTICAL PROGRAMS & GAMES IN BASIC £6.65
Programs for everything from Space war games to Blackjack.

Waite. M. MICROCOMPUTER PRIMER £6.40

Waite, YOUR OWN COMPUTER £1.60
Introduces the beginner to the basic principles of the microcomputer.

Ward. MICROPROCESSOR/MICROPROGRAMMING HANDBOOK £6.00
Authoritative practical guide to microprocessor construction programming and applications.

Veronis. MICROPROCESSOR £13.00

Zaks, R. INTRODUCTION TO PERSONAL AND BUSINESS COMPUTING £8.60

Zaks, R. MICROPROCESSORS FROM CHIPS TO SYSTEMS £8.30

Note that all prices include postage and packing. Please make cheques, etc., payable to Computing Today Book Service (Payment in U.K. currency only please) and send to:
Computing Today Book Service,
P.O. Box 79, Maidenhead, Berks.
Prices may be subject to change without notice.

A.P. Stephenson.

After the first mad programming spree with your machine you may like to build yourself a library of useful programs.

The first few weeks after the purchase of a home computing installation may justifiably be defined as the "infatuation" stage. The power of the machine to generate data at apparently pheomenal speed is fascinating, even exciting to those new to the computer keyboard. Scores of little programs are lovingly saved on cassette tapes most of them centred around the FOR/NEXT loop. Typical programs include printing out "HELLO" 47 times, filling the screen with nine-digit columns of sin(x) and −cos(x) or meaningless equations chosen primarily for their complexity. As many of these little morsels as possible are crammed on both sides of C60 (or in some cases even C120!) tapes. Frantic trips to purchase new supplies of blank cassettes are frequently made or, if the shops are shut, a previously loved recording of Beethoven's ninth is irrevently erased in order to make room for a program which generates the first 2000 primes (I often wonder what you do with primes after you generate them but they seem to offer solace to many).

Naming Names

But all things come to an end at some time or another. It gradually dawns on most people that their "collection" is in reality nothing more than a heap of rubbish. Most of what they have saved is useless, and the few that have some merit are buried between dozens of unwanted remnants.

Organisation, The Key?

Any attempt to organise your computing life must begin with a simple rule. . .one program on a tape with a copy on the reverse side. Superficially, this appears to be a shocking waste of tape because, on the average, most of the tape will remain unused but in spite of this the rule is sound in human terms. It is better to waste a few feet of relatively inexpensive tape in return for the following benefits: no infuriating searches for programs "in the middle"; no need to name programs and therefore no need to memorise what you have named them; if you have to amend a program, there is no danger of the extra few bytes extending into and obliterating the beginning of the next program; if the tape is accidentally dropped into a plate of soup (or similar household hazard degrades its performance) only one program is lost; if you lend a tape to a friend for copying purposes and it is returned a corrupted length of jargon, there is less danger of physical violence breaking out if only one program is spoilt. Finally, we cannot entirely discard a psychological factor. Weeks, perhaps even months of programming work condensed onto one tape fails to impress the casual aquaintance. Spread out into twenty or so, neatly labelled cases with the whole resting in a partitioned "cabinet" will enhance your local reputation as an egghead. There is one nagging doubt which must remain to PET owners . . . why did Commodore take such pains to provide a truly magnificent tape handling, program-naming facility if the foregoing advice is taken? Perhaps they just failed to appreciate how easily the normal human being looses patience. Those, lucky enough to afford a floppy disc system will of course have no need for this advice; the facility to name programs on a disc is as essential as it is unessential on tape. One final word on this matter. . . buy only C12 tapes, or less than C12 if you can get them.

Worthwhile Programs

"Worthwhile" in this sense means "is it worth saving on tape?". Consider the following as a reasonable set of criteria from which to start:

1) Has the program been tested for every conceivable input combination. For example, what happens if you input a "0" or a negative number or a number with umpteen digits in it? Nothing is more humiliating to a proud demonstrator than one of those sarcastic error messages which leap up from the bowels of the BASIC interpreter whenever it suffers the slightest confusion. Particularly if you are trying to impress.
2) Will the program check for ridiculous input? Remember that an input can be mathematically acceptable and free from syntax error but can still lack realism. For example, let us assume a program, which assists in the design of a signal amplifier, asks for the supply rail voltage. If the operator mistakenly keys in 2.6E4 instead of 2.6E−4 will the stupid machine accept this. . .or what is more to the point. . .will the stupid program accept it and go on to compute a recommended output current in the order of kiloamps? In short, does the program include full data input validation routines?
3) Is the program reasonably crash proof. This calls for considerable effort and it is not always possible to achieve a 100% confidence factor.
4) Is the program completely self-explanatory to the operator? Are there for instance, full instructions on the VDU screen or does it mean searching for some scrap of paper somewhere which contains the gory details on the button-pressing routines? No accompanying document of any kind should be necessary because the VDU screen can tell all. There should also be a title page which defines clearly the purpose of the program. Remember that at the time of writing, the purpose is all too clear but after a few weeks or months the memory fades.
5) Is the textual material on the VDU easy to understand and pleasantly arranged? There is no excuse for sloppy presentation and curt chunks of computer jargonese interspersed with abbreviations. Just because the computer has no soul or manners this is no excuse for omitting the pretence. A little care taken in presentation will give the pleasant illusion that lurking behind the cold rectangular sheet of glass is a "being" with a heart of gold . . . kindly and paternal when the occasion warrants it and yet no hesitating to deliver streams of pure vitriol if its human operator enters silly figures or presses wrong buttons. In other words, give your computer a personality. Space out the text in a readable manner. Nothing is more tiresome than a page full of

closely spaced reading matter, particularly if it is composed entirely of capitals. If your computer has no lower case letters, make up for it by spreading out the material with line feeds between. There is no need to stuff everything on one VDU page but never allow the pages to scroll. Text creeping up from the bottom and disappearing at the top should never be tolerated; it is unpleasant to read and amateurish. The most cardinal sin of all is to allow words to be bissected, the second half wrapping around onto the next line. A final piece of advice concerns erasing the screen frequently. Never allow bits and pieces to hang about such as relics from the operating commands. When for example you type the command RUN, ensure it is wiped off by a "clear-screen" statement immediately. It is a horrible mess to see past records of fumbling operating scattered about the screen and polluting the program text. The message "PRESS PLAY AND RECORD" is ideal in its proper place but once you have pressed the buttons it has about as much relevance as last week's football results. Rub it off.

6) Is the program planned with the idea of future expansion or improvement in mind? No program can ever be perfect and equally true, no program can ever be absolutely complete. There will always be the nagging doubt, particularly when it is re-run a few weeks later, that some extra facility or twist should have been added. In many cases however, this can be a difficult or even impossible task. In the first case, the program may be utterly incomprehensible when LISTed if several weeks have elapsed since it was written. Juggling with obstinate statements, temper, frustration and the other multitude of ills popular during program construction eventually leads to a transient state of euphoria when the beast finally decides to work. There is a mad rush to "get it on to tape" and indulge in a satisfying bout of self-congratulation. It takes a little while to appreciate the value of the REM statement because at the time, it seems unnecessary. In fact some of us deliberately leave out remarks in order to prevent other people understanding how our masterpiece works. This attitude can be self-destructive because the writer of the program may eventually become the victim. Another obstable to future amendment is a poorly structured original and close-packed line numbers. Never start a program with line number less than 100 in case some extra stuff may have to be squeezed in at the head. Be methodical in the choice of subroutine line numbers. Stick them all together well down the bottom, say at line 10,000 onwards. In this way, you will avoid the ugly embarassment of having to leap frog over them with a wasted GOTO statement when the lines start to creep down further than the original estimate allowed. The term "program structure" of course means a lot more than the mere organisation of line numbers. It means laying out a program in neat little modules, each capable of being individually tested in its own right. In fact there is a specific programming philosophy with many little rules and regulations resting beneath the blanket title of "STRUCTURED PROGRAMMING". This is worth detailed study if only to know when to break some of the rules.

Programs To Write

Advice on what programs to write is about as difficult as advising on the best length for a piece of string. An overall piece of advice is simply to walk before you run. Don't attempt to write wildly ambitious programs unless you are quite certain you understand the full implications of the task ahead. Unfortunately, it takes some experience to know in advance whether or not a certain programming task is likely to be easy or horribly difficult; computers are odd things. For example, if someone came and asked me to write a program to print out a table of the singular solutions of a second order differential equation I would take the money in advance and probably deliver the goods (suitably tarted up in accordance with the previous advice) the next day. This is not because maths and physics is my strong point (I might pass 0-Level maths with difficulty) but because the actual maths details must reside in some text book equation somewhere or other. It would just be a case of letting the faithful old BASIC interpreter handle the sordid details once the correct sequence of brackets and operators have been entered from the text book to the VDU. Such programs are elementary number crunching exercises, impressive but routine. On the other hand, a request for "a little program to sort and classify my butterfly collection" could turn out to be a nightmare. The following is a crude attempt to group the classes of programs which can be written and appropriate remarks on their respective difficulty factors.

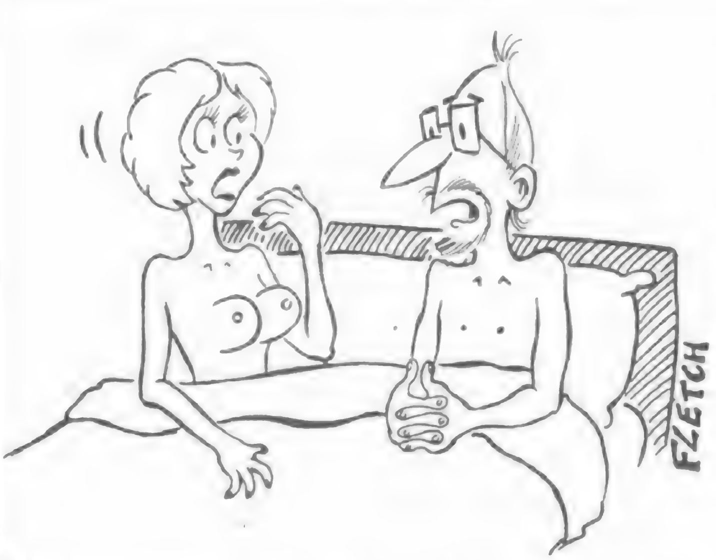
Numbercrunching. These follow a relatively simple pattern; inputting the required parameters, fitting them into the "equation line" and displaying the results in a clear manner.

Two subroutines should be considered almost indispensable to number-crunching activities, one to round off numerical results to a desirable number of decimal places and the other to line up the decimal points. Answers like 34.5689302 inches or £67.24578945 lack realism and the sight of a VDU screen full of figure groups zigzagging from top to bottom is not only difficult to read, it is quite revolting in appearance. Always use TAB(n) to position volumns, the semicolon as a delimiter encourages zigzagging. I find it curious that the BASIC software writers decided to invoke the exponent form of print out for numbers smaller than 0.01; it seems far too "early", because we are used to seeing numbers this small in every day life and smaller still in science.

Keyboard quizzes. Many sophisticated programs have been written under the general title of Computer-Aided-Teaching or Computer-Aided-Learning. Less ambitious but surprisingly useful programs are relatively easy to write (and certainly worth saving) based on questions and answers. The set of questions and answers can be in a group of DATA statements, and called up under two READ statements. The first will call up the question and the keyboard response is checked for correct match with the second, which is the "answer".

An extra twist is to incorporate random selection of the pairs to stop the operator using a sequence. There is however, an element of danger in this type of program. It tends to breed quiz addicts. Tape after tape is saved on all possible subjects until the entire household takes on the appearance of a Bamber Gascoigne Show.

Games. This area is undoubtedly popular and it cannot be denied that senior programmers in the professional classes devote many hours to thinking up new games or introducing new twists to existing ones. Unfortunately, a game program, unless particularly novel and interspersed with exciting animation takes a disproportionate time to program in relation to the subsequent playing time. As programming exercises they are superb. Whether many of them are really



"FRANKLY MISS WITHERSPOON, I'M CONSIDERING REPLACING YOU WITH A PROGRAM LIBRARY..."

worth the tape storage is debatable. Consider for example the class of games which could be covered by the classification "Moon Landing". They all follow the same well-worn path . . . .you are in some dangerous James Kirk situation . . . too much throttle and you run out of something or other . . . too little and you crack the surface of the moon or Mars or whatever particular member of the galactic regions happens to fit the title. They will all contain a couple of equations from the Newtonian tables, suitably embellished to fit the game. The most awkward thing to get right in programming such a game is the difficulty-factor. Too hard and the player is frustrated; too easy and the game is described as boring.

The behaviour pattern of the players, however much care is taken with the programming details, is distressingly familiar. Great enthusiasm at first but declining exponentially towards complete apathy. For those who have a genuine love for game programming the following little tips may be found useful :

Explain the rules concisely in the title page.

Display as much animation as your skills in programming allow.

Don't allow the computer to respond "too instantaneously". An apparently immediate response does not impress the player.

Choose your GO–BACK–TO destinations carefully. It is pleasant for the ego when the computer asks for your name and instantly promotes you to "CAPTAIN. . . . ." but it soon becomes an irritating chore if this ritual has to be repeated on each replay.

Take particular care to make programs crash-proof. There are some who, finding themselves in an irretrievable position, would crash the program rather than suffer the humiliation of being beaten by "some damned machine".

Try and add a few original twists. For example, allow a few loop holes for cheating but make the computer respond with something like,

> "We noticed your pathetic attempt
> at subterfuge three lines ago but in
> view of your obvious immaturity, we
> decided to overlook the matter. Should
> it occur again you will be disqualified."

Note the use of the royal "WE" above . . . very useful little dodge to create an air of omnipotence, although don't overdo it by using phrases like "My RAMS and I . . . .".

Dynamic Art. Providing the word "art" is not taken too literally, some quite astonishing moving patterns can be generated on most of the home computers. They are however far more impressive if you are fortunate enough to own an APPLE or other model which includes colour combined with high-resoltuion graphics. The PET, despite the great play made of its "graphics facility" is not really suited to the job. It certainly has very useful graphic "keys" but the resolution in general is pathetic; equivalent to painting a portrait with a ten inch ceiling brush.

Sorting, organising and retrieval of DATA. It is this area that the computer is truly at home. Every home computing enthusiast should take "data processing" seriously. Strange how so many writers attempting to teach this subject use examples like milk bills to start off with. Milk of course is a delightful source of health giving energy but the compilation of milk bills is not likely to cause a flutter of excitement, followed by a mad rush to write the program. My wife would look at me in sheer astonishment if I suggested she used my PET each month. She would probably write it out on the back of an envelope in ten seconds flat, certainly before I would have time to fumble round the back for the ON/OFF switch. It is appreciated of course that such simple examples are typical weapons of the educationalists, based on the principle "teach from the known to the unknown", "use homely analogies" etc etc.

There is a danger however of de-glamourising a subject and underestimating the public mood and intelligence. Why not substitute plutonium imports for milk bills? The program would be just as easy to write and marginally more exciting.

Tape books. Sales brochures often draw attention to advantages of storing useful day to day information on home computers, recipes etc. General purpose reference "books" can certainly be very useful on tape, providing there is a title selection page or pages. Once the tape is loaded (the most annoying stage), it is quicker to get at a given page by pressing a number key than turning the pages of a paper book. The floppy disc is naturally the ace peripheral in this field but, alas, still a little pricey for the likes of us.

Programming

It is difficult to say anything original on this subject. Literally hundreds of books have been written on the BASIC language alone, besides the thousands written on programming principles in general. However good the manuals supplied are it is almost essential to dip into the pocket again and buy at least one book on BASIC. Which one? For what it is worth, I have been impressed (and educated) by "BASIC AND THE PERSONAL COMPUTER" by Thomas A. Dwyer and Margot Critchfield but there are probably dozens of others equally as useful. The following little snippets of wisdom (?) may be of some assistance to those who, like myself, have no *natural* abilities in the art of programming.

1) Buy a good book on BASIC and carry out *EVERY* example in it. It's not a bit of use just "reading" a book on this subject.
2) Buy as many magazines on computing as you can afford, in addition to this one of course.
3) Keep a notebook, or preferably a card index system, and copy down every little programming "module" or dodge which has general purpose use. In this way you gradually

acquire a background in fundamental techniques and you can slip them in your programs whenever the need arises. Is this cheating? Depends on how you define cheating. There is little point in re-inventing the wheel on every possible occasion. Isaac Newton, not renowned for his modesty, once replied to a remark by an admirer, "If I have seen a little further than most, it is because I have stood on the shoulders of giants". To copy down a complete program and pass it off as one of your own is of course a different matter. Ethics apart (not particularly fashionable nowadays anyway) some one else might have read the same magazine and bang goes your reputation! The sort of modules worth saving for future and continuous use include, lining up decimal points, rounding to n significant digits, sorting numbers into ascending or descending order, sorting names into alphabetic order, etc etc. A word of warning regarding program modules or indeed full programs printed in magazines. Some of them don't work! The usual cause is a misprint somewhere along the line and readers, to judge from the rather acidic tone of their letters, express surprise that "the Editor doesn't proof read them before printing". Proof reading costs time and money for normal kinds of text but to proof read computer programs to guarantee 100% error free would probably treble the cost of a magazine. In any case, if they don't work then make them work . . . it's good practice anyway and the mistake is often the trivial omission or incorrect insertion of a comma or quote or perhaps unmatched parenthesis.

4) Produce a tape of useful subroutines based on the previous suggestions and load in this tape as a matter of habit *before you start on any program*. Make sure that every subroutine has an explanatory REMark which defines the parameter variables. As an example,

```
10000 REM***ROUND N8 TO D8 DECIMAL PLACES***
10010 N8=INT(10^D8*N8+0.5)/10^D8
10020 RETURN
```

Why choose such a strange variable (N8)? Precisely because it is strange and therefore unlikely to have been used in the main program. A question arises after the subroutine tape has been loaded. . .suppose all of them are not used? It doesn't really matter because an unused subroutine can do no harm. If memory space becomes critical then naturally erase the unwanted residue.

5) Join a local computer club. They tend to be friendly gatherings all anxious to learn from each other and refreshingly free from professional snobbery of any kind. The home computer addict tends to be though of as slightly weird by "normal" people, a kind of mutation. It is comforting to spend a few hours in the evening with other mutants. The great thing is to join soon while the hobby is still young. As the numbers of these clubs grow and the membership expands to excessive limits, the character may change. It could reach a state like that which exists in the so-called "exclusive" golf clubs, questionnaires on various aspects of the applicants background. Perhaps, God forbid, they may even require that supreme emblem of respectability, the club tie!

The Final Words

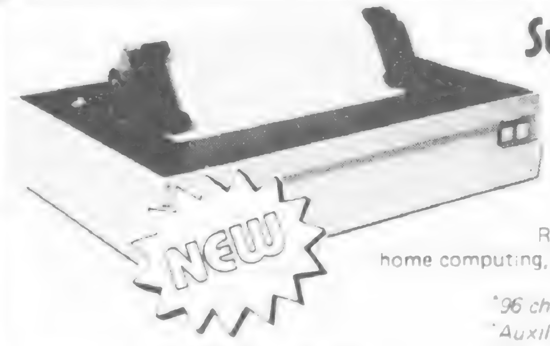
In conclusion, it is worth examining some advice given in the manuals concerning the art of programming. Apparently, it is a cardinal sin to compose at the keyboard . . . it is called "winging it". We are instructed by the tribe elders to write the complete program on paper before approaching the keyboard; at least every separate module. This discipline came into being because of two non-related influences. Firstly, the influence of the academic purists who insist on a carefully thought out logical approach on paper first. The second influence was that of practical necessity. Prior to the microprocessor and high density integration of semiconductor memory, computing was very expensive, VDUs were non-existent or rare, every response was spewed out on reams of expensive paper and, above all, the cost per minute precluded the luxury of idle doodling.

The position with the home computer is different. Very few of us can afford printers anyway. . .at least not in the first year of ownership. The VDU wastes nothing. It is a perfect doodling pad and unlike paper, can be used over and over again. It is, however, a good idea to draw out a rough plan of campaign in the form of an outline flowchart, prior to operating the keys.

Another discipline carried over from the past is an obsession with memory economy. It seems pointless to prune a program (that works) down to the last byte unless there is a real danger of running out of memory. If you have say, an 8K memory and your unpruned program takes 6K why fiddle about with it. Tricks like multiple statements per line to save a line-return byte are admirable when the necessity arises but the subsequent readability is poor. To increase execution speed just for the sake of it is another pointless operation. If your program works and it is reasonably "tidy" leave it alone and get on with another. In this way your tape library will grow much quicker and be just as useful as those of your fusspot colleagues.

" DO YOU HAVE 'CROSSROADS' ?"....

Super Print 800



80 COLUMN HIGH PERFORMANCE IMPACT PRINTER

The ideal companion for PET, Apple, TRS80, Exidy, Superboard, Compukit and most Micro's

Rugged metal enclosure makes it ideal for home computing, small business systems, data logging etc.

Just look at these standard features

- *RS–232, 20mA, IEE 488 and Centronics 1.0*
- *16 Baud Rates to 19,200*
- *60 Lines per minute · Bidirectional*
- *5 print densities 72,80,96,120 or 132 Chr Line*
- *Self Test Switch*

Standard Feed-Model 800
ONLY £329 + VAT

- *96 character ASC II Standard*
- *Auxilliary User Defined Character Set*
- *Accepts 8½" max. paper · pressure feed 9½" max. paper · tractor feed*

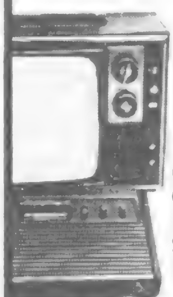
OPTIONS:

- *Tractor & Fast Paper Feed Graphics (model ST)*
- *2k Terminal Buffer*

Tractor & High Speed-Model 800ST
ONLY £389 + VAT

Atari VCS System

Most popular top range micro-video game



Current carts. include · Air-Sea Battle, Space War, Outlaw, Video Olympics, Breakout, Basketball, Surround, Blackjack, Basic Maths, Hunt & Score, Miniature Golf, Skydiver, Street Racer, Chess, Backgammon and many more

Cartridges mostly · **£13** + VAT
Chess (8 levels) · **£39** + VAT
Backgammon · **£30** + VAT
Superman · **£20.80** + VAT
Indy 500 · **£30** + VAT

Coming Soon — BASIC Computing, Space Invaders plus many more !
Buy a VCS, introduce a friend to buy one and get a Keyboard FREE worth **£19** (limited Ingersol Offer)
Atari VCS NOW **ONLY £113** + VAT + your first cart. · **£6.90** + VAT (worth **£13**)

Nascom-2

MICRO COMPUTER

ONLY £295 + VAT

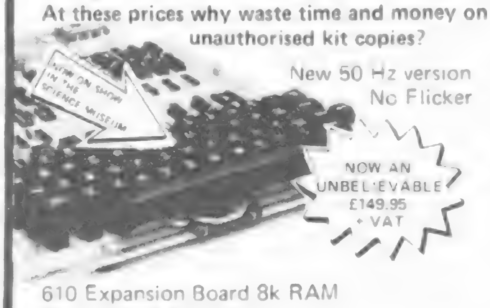
FREE 16k RAM Board

Z80A 8 bit. This will run at 4 Mhz but is selected between 1/2/4 Mhz.
On-board, addressable memory. 2K
2K Monitor—Nas-sys 1. 1K Video RAM (MK4118). 1K Work space/User RAM (MK 4118). 8K Microsoft Basic (MK 3600 ROM). 8K Static RAM 2708 EPROM

Superboard II - OHIO - Challenger 4P

At these prices why waste time and money on unauthorised kit copies?

New 50 Hz version
No Flicker

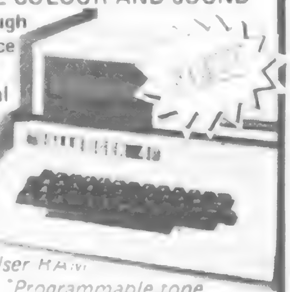


610 Expansion Board 8k RAM
ONLY £149.95 + VAT
IP CD3P Minifloppy Disc, Cased, PSU, 2 copies of DOS · **ONLY £289** + VAT
Set of 4k RAM (Superboard users only)
ONLY £24 + VAT
Plastic Case · Beige **ONLY £26** + VAT
Challenger IP · Metal Cased, Superboard, PSU modulator · **ONLY £188** + VAT
Discs & Cassettes available

PAL COLOUR AND SOUND

A major breakthrough in price/performance for personal/ business educational micro's

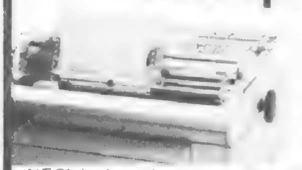
Single RF O/P provides both sound & vision to colour TV. (Colour sets only)



8k Microsoft 8k User RAM 32 rows x 64 cols. Programmable tone generator and DAC. Joystick and keyboard interface

Challenger 4P · 8k Basic in ROM 8k RAM expandable to 32k · **ONLY £425** + VAT
Challenger 4P MF · Same as 4P but with Mini-floppy 24k RAM, expandable to 48k
ONLY £950 + VAT
Complete range of accessories available

NEC Spinwriter



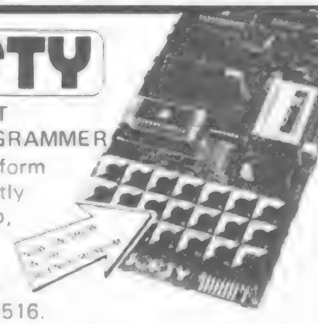
- for the professional word processing system

£1795 + VAT

NEC's high quality printer uses a print "thimble" that has less diam.and inertia than a daisy wheel.Giving a quieter, faster more reliable printer than can cope with plotting and printing(128 ASC II chs.) with up to 5 copies,friction or tractor fed. The ribbon and thimble can be changed in seconds. 55 chs.per sec bidirectional printing-with red/black,bold,subscript, proportional spacing, tabbing, and much much more

SOFTY

INTELLIGENT EPROM PROGRAMMER



Built or in Kit form
Connects directly to TV. Develop, Copy, Burn, Verify 2708, 2716 and with modification 2516.

Softy is a versatile product and each application will be different by definition. When Softy is connected via a serial (RS 232) or parallel link with any small computer capable of supporting an assembler a simple and capable Product Development System is formed. For product developments with less than 2k of firmware Softy may be the only development tool you will need.

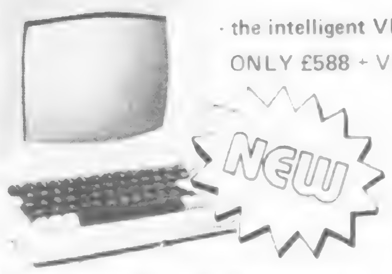
- *Fast cassette interface-over 2000 baud eqiv*
- *Ideal Training Aid-useful control computer*
- *Equivalent dev. systems cost over £500+*

ONLY £120 + VAT Built & Tested
£100+VAT Kit. £20+VAT Built Power Supply

miniMAS Terminal

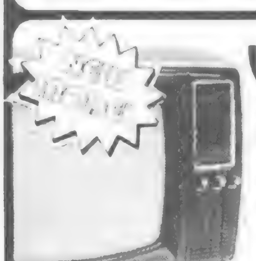
· the intelligent VDU

ONLY £588 + VAT



Ulitises the new advanced and powerful Z8 Micro. 12" P4 Phosphor high resolution display. 24 lines of 40 or 80 characters - 25th line status. 9 x 13 field displaying 128 ASC II characters · 4K RAM · 2 page. RS232C or RS432 · 16 independent baud rates for each I/O. Full 92 keyboard - sophisticated cursor controls. Emulates DEC VT52

Video 80



- based on probably the world's largest selling video monitor -over 100,000 in use !

ONLY £69 + VAT

Ideal for home, personal and business computer systems. 12" video display. Composite video input. Compatible with many computer systems. Solid state circuitry. Stylish rugged plastic housing

Mighty Micro

In association with Watford Electronics

Please add VAT at 15%. Carriage extra, will advise at time of order. Official orders welcome. Product details on request. Trade and export enquiries welcome.

33 CARDIFF ROAD, WATFORD, HERTFORDSHIRE

Tel: (0923) 40588/9 Telex 858747. Open 9 am to 5.30 pm Mon - Sat.

Close to Watford Football Ground. Nearest Station - B.R. Watford.

BUY IN CONFIDENCE — In the event that we are unable to deliver your goods within 7 days,we do not bank your remittance until such time as we have the goods to despatch. If on receipt of your order the goods do not meet with your satisfaction, return within 7 days for full refund.

HP85 BRIEF

Henry Budgett.

At last a 'real' computer, from the people who brought you the first scientific calculators. This preview will shortly be followed by an indepth review.

Under normal circumstances the person who is reviewing a system for the magazine has about a fortnight to play around with the machine and discover its weak points. I say under normal circumstances because in this case that rule does not apply. Reviewing the HP 85 was almost a nostalgia trip for me as I have used their desktop minis in the past and there is a large degree of similarity. However this is a personal microcomputer, a very different collection of chips to your average "home computer" not only because it costs around £2000 but because it has been designed as a professional tool for research and development, laboratory and educational establishments. Why have we reviewed a system that is probably outside most people's financial resources? The obvious answer is that of general interest, the better answer is that this system shows how much can be done by a micro — basically it makes most "home computers" look like overgrown video games and I'm not putting them down either.

The Total Specification

The HP 85 is based around a custom built chip set. The eight bit CPU, the dynamic RAM controller and the I/O controllers are all designed for the job by Hewlett Packard and as a result the whole CPU board contains only sixteen ICs. The system is built into a smart Apple-like console but is about 8" high to include the five inch VDU, the cartridge tape drive and the thermal printer mechanisms.

The first section that makes an impression on you is the keyboard. Not only have you a full ASCII set and a numeric pad but there are full editing keys; including HOME, cursor controls, insert/replace, line delete, character delete and more besides. In the top left corner under the VDU is another row of mysterious keys, these are user programmable under BASIC to perform specific functions within the program. I shall discuss the BASIC at a later stage in this review, it really deserves a separate article. Also available on the keyboard as direct command keys are functions such as LOAD, STORE, COPY LIST, PLIST, RUN, PAUSE, STEP etc etc that you use instead of having to type in the normal BASIC command.

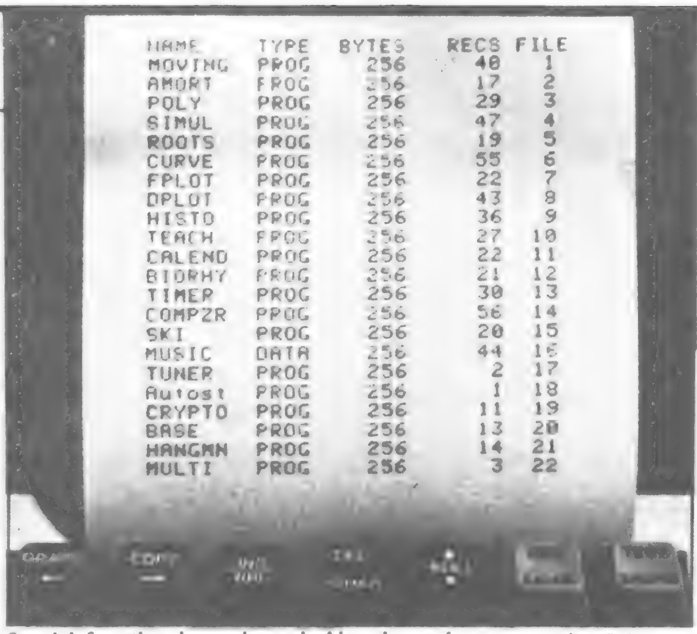
The VDU is a five inch monitor, rock steady, with an independent memory of four screens full that can be re-accessed by a ROLL key. The format is 32 characters by 16 lines, a program "line" can be up to 95 characters (3 lines minus 1) long. The graphics capability of this screen is very impressive, as the photos hopefully show, and you have program access to 256 by 192 dots or a total of 49,152 to save you working it out in your head. The graphics capability is so powerful that we are going to cover it in a future article!

Mass storage is dealt with by a tape cartridge, *not* a cassette, that is specifically designed to do digital data storage. Unfortunately the system is not completely compatible with the desk-top minis even though the same physical format is used. However it is a damn sight better than any cassette system and it is much faster. The speed improvement is achieved by making the thing emulate a soft sectored floppy, it reads a directory, finds the location of the program

and then spools off to load it. Commands are simply LOAD "FRED" where LOAD is a single key anyway, STORE "FRED" where once again STORE is a single key, and a couple of other commands for securing programs or data and conditioning the tape. There is a special tape file called "Autost" which is automatically loaded and run if there is a tape in the transport at power on. On both of our sample tapes, one of which – the Standard Pac – is supplied with the machine, this program was a little graphics routine. However it is a simple matter to install a program that actually does something useful and this is where the HP comes into its own as an instrumentation controller.

A BASIC program using just some of the 85s extra commands.

```
 10 REM *MULTIPLICATION PROG
 20 REM *THIS PROGRAM WILL PRODU
    CE
 30 REM *MULTIPLICATION TABLES F
    OR
 40 REM *ANY INPUT NUMBER, N
 50 REM *UP TO AN INPUT LIMIT, M
 55 REM *INTEGERS ONLY!!
 60 REM *OPTIONAL PRINTER OUTPUT
 70 CLEAR
 80 DISP "INPUT YOUR REQUIRED NU
    MBER (0<N>100)"
 90 INPUT N$
100 IF VAL(N$)<=0 OR VAL(N$)>100
     THEN CLEAR @ GOTO 80
110 N=INT(VAL(N$))
120 CLEAR
130 DISP "INPUT YOUR MAX VALUE (
    0<M>100)"
140 INPUT M$
150 IF VAL(M$)<=0 OR VAL(M$)>100
     THEN CLEAR @ GOTO 130
160 M=INT(VAL(M$))
170 CLEAR
180 DISP "PRINTER OR DISPLAY P/D
    ? (DEFAULT TO SCREEN!)"
190 INPUT R$
200 IF R$#"P" THEN 270 ELSE 205
205 PRINT TAB(5);"* THE ";N;" TI
    MES TABLE *" @ PRINT @ PRINT
210 FOR L=1 TO M
220 PRINT USING 320 ; N,L,N*L
230 NEXT L
240 PRINT @ PRINT @ PRINT
250 CLEAR
260 GOTO 10
270 CLEAR @ C=0 @ REM SET UP SCR
    EEN COUNT
271 FOR L=1 TO M
275 IF C#15 THEN 280 ELSE C=0
276 WAIT 5000 @ CLEAR
280 DISP USING 320 ; N,L,N*L
286 C=C+1
290 NEXT L
300 WAIT 10000
310 GOTO 10
320 IMAGE 3D,2X,"TIMES",2X,3D,2X
    ,"=",7D
```



Special function keys abound. Also shown is an example of the thermal printer output.

The thermal printer is a 32 character-per-line, bidirectional device which can handle all the HP graphics and characters. A hard copy can be generated *at any time* by the command key COPY or this can be executed as a program statement. Under normal circumstances the VDU is the default device but the BASIC command PRINT refers to the PRINTer so in the sample programs you will see DISP for VDU access. This status can be reversed by a command or you can tell the system to PRINT ALL – which it promptly does. The paper supply is generous to a fault, I started off with less than a full roll and in three weeks of printing everything possible I still haven't got down to the end. The machine always prints graphics displays the wrong way round, this is done so you can have endless strip charts – you can even print a musical score. Talking of music there is a programmable bleeper that renders an acceptable version of "William Tell" but has a more functional use as a warning device.

The Language Barrier

The HP 85 has built in BASIC, but here again the similarity with other systems ends. The language and the operating system are built into 32Ks worth of ROM with the bottom 8K being stackable. The language exceeds all current ANSI standards and even a glance at the sample programs will reveal that it is more than a little different to the usual Microsoft versions. A few of the more unusual commands and functions are given in the programs but we are going to devote a section of the follow-up article to the language. The capabilities of a programmer used to a simple BASIC such as Integer, or even Extended, will not even be touched by this

```
* THE  12  TIMES TABLE *
12  TIMES    1  =      12
12  TIMES    2  =      24
12  TIMES    3  =      36
12  TIMES    4  =      48
12  TIMES    5  =      60
12  TIMES    6  =      72
12  TIMES    7  =      84
12  TIMES    8  =      96
12  TIMES    9  =     108
12  TIMES   10  =     120
12  TIMES   11  =     132
12  TIMES   12  =     144
```

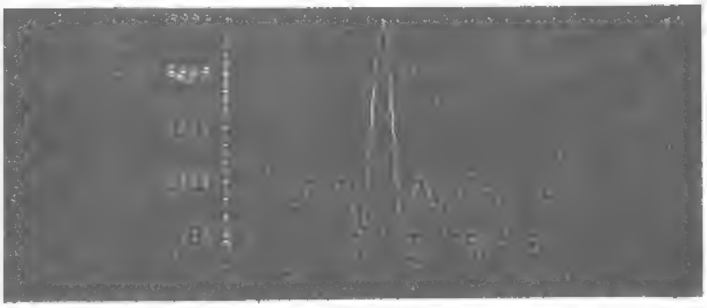
HPs cryptography program, good fun!

implementation. Your frustrations at not being able to solve that problem on your home system disappear at a stroke with this machine. Although machine code is not yet available for the 85 the only reason you are ever likely to want it is because of the speed factor. Not that I'm inferring that the BASIC is slow — it isn't — but rather that you can do so much with it, it would be nice to pass the routine sections to machine code. Perhaps I'm being just a little too much of a purist, I certainly never found the need in my investigations.

Another superb set of facilities that come under the heading of "Utilities" are built into the machine. These include automatic line numbering, re-numbering and a whole bunch of diagnostics and debuggers. As a measure of the thought that has gone into the system the re-number command checks to see if the standard — or your quoted — start and displacement values will cause an overflow. If this is the case it starts at line 1 and works on in steps of 1 —automatically!

The debugging tools include single step, tracing of all, or specified, sections, full error code display with descriptions and more besides.

```
10 DIM I$[32],F$[1],M$[2000]
11 CLEAR
20 DISP "CODE OR DECODE: C OR D
   "
30 INPUT F$
40 IF F$="C" THEN L=1 ELSE L=2
50 DISP "CODE NUMBER PLEASE"
60 INPUT S
70 RANDOMIZE S
80 M$=""
90 DISP "TYPE MESSAGE ONE WORD
   AT A TIME.  TYPE '*' TO END
   MESSAGE"
100 DISP "GIVE ME YOUR MESSAGE"
110 INPUT I$
120 IF I$="*" THEN 160
130 ON L GOSUB 1000,2000
140 M$=M$&C$&" "
150 GOTO 110
160 PRINT M$
161 DISP "CONTINUE ?"
162 INPUT A$
163 IF A$="Y" THEN 10 ELSE 170
170 END
1000 REM *ENCODING ROUTINE
1010 C$=""
1020 FOR I=1 TO LEN(I$)
1030 C$=C$&CHR$(65+(NUM(I$[I,I])
     +INT(26*RND)) MOD 26)
1040 NEXT I
1050 RETURN
2000 REM *DECODING ROUTINE
2010 C$=""
2020 FOR I=1 TO LEN(I$)
2030 C$=C$&CHR$(65+(NUM(I$[I,I])
     -INT(26*RND)) MOD 26)
2040 NEXT I
2050 RETURN
```



A small example of the graphics capability on the 85.

Additional Firmware And Expansion

As I mentioned earlier the bottom 8K of ROM can be stacked. This means that firmware packages of up to 8K in size can be nested over this section of memory and run instead. This is in addition to any firmware that you may wish to locate in the spare 16K of memory. If you wish to use the expansion for RAM then this is achieved by plugging a special drawer into the bus slots at the back of the case with an extra eight RAM chips and a controller chip. The bus is called Capricorn, but there are adapters for RS232 and the HP/IB or IEEE-488 bus currently under development. This means that the 85 will, just like its bigger brothers, hook into all the fantastic range of instrumentation and peripherals that are available : plotter, line printers, atomic clocks etc etc.

Discs are not yet available but are under development, after all they are only slightly different to the existing minicomputer discs. About the end of the year is an expected date.

Philosophy

Just what is the 85 capable of, and why produce a micro when there are already minis available to do the same job? Well the simple answer is that the 85 is a cheaper system than the minis, it can do the same job but it's a bit slower. I suspect that many people will try to use it for a small business machine, or a general purpose system but, while this is not wrong, it is rather an underuse of a computing machine.

The price is definitely a subject for controversy, after all the US price works out to around £1200 and that is suspected of being too high. The simple fact is that the price is artificially high, the reason being that HP probably can't make enough of them. It is strongly rumoured that the entire year's production was sold within a matter of days of the launch, bad news for people who want one after reading this review!

In Conclusion

The HP 85 represents what is almost certainly the first of a new generation of micro computers that compute. The age of overgrown video games and simple systems has been surpassed, at very little increase in price, by a design that is unashamed to be a computer. I suspect that within a year the price will have fallen to around the £1200 mark and it will become a widely available machine. However if this had occurred at the launch, just a few weeks ago, the market in personal computers would have been sewed up overnight by HP.

Whilst the system represents superb design, engineering and support the price is simply too high for the home user at the moment, unless he or she is well off. The quality of the documentation, and the supplied software is of HPs usual excellent quality, although not aimed at the novice, and it even has one games program!

interface components

NASCOM IMP AVAILABLE EX-STOCK

The incredibly low-price Nascom IMP is now available off the shelf at Interface Components. It plugs into any microcomputer system with a serial RS 232 interface, including the popular Nascom 1 & 2.

Although an impact matrix printer, its versatile feed mechanism allows it to accept A4, foolscap and quarto letterheads making it suitable for word processing applications. And it's quiet too. Line printers and many typewriters are deafening by comparison.

Finished in a smart blue plastic enclosure, the IMP is only £325 plus VAT (post and packing extra) completely assembled.

FEATURES

- 80 characters wide
- 60 lines per minute
- Bi-directional printing
- 10-line print buffer
- 96 character ASCII set (includes $,#,£)
- Automatic CR/LF
- $8\frac{1}{2}''$ paper (pressure feed)
- $9\frac{1}{2}''$ paper (tractor feed)
- Baud rate 110 to 9600
- External signal for optional synchronisation of baud rate

ORDER FORM

Send your orders to:
Interface Components Ltd, Oakfield Corner, Sycamore Road, Amersham, Bucks HP6 6SU.
Telephone: 02403 22307 Telex: 837788.

Please send me.........Nascom Imps at £325 each plus VAT at 15% plus £2.50 p&p.

Name

Address

Access or Barclaycard No.

Personal callers welcome

CT/4/80

Sixteen pages of readers own program submissions for your enjoyment.

The next sixteen pages are packed with useful programs, games and ideas, for a variety of systems. All have been sent in by readers for our Softspot feature, and because of this they are untested by us.

If you think your software is of good enough quality to be included, send it to the following address :

CT Softspot,
Computing Today,
145 Charing Cross Road,
London WC2H 0EE.

and we'll take a look. We pay for all accepted programs, if you want yours returned please enclose an SAE. We do not demand line printer output, fine if you have one, all we ask is that the manuscript is legible and includes diagrams and explanations where necessary. A full (written) listing is mandatory.

So, for the next few pages, it's all yours!

J.F.Kendall.

TI59 ROUTINES

Anyone who is in the position of programming a microprocessor in machine code will have surely found calculating in hexadecimal rather tedious, to put it mildly, especially in the case of negative displacements for relative jumps where the two's complement has to be found. Hence this program was written to make hex. working much easier.

It incorporates the following features :–

(1) Converts hex. to decimal and vice versa.
(2) Finds two's complement of a hex. number.
(3) Input any two hex. addresses and it will output the signed hex. displacement index for a relative jump from the first to the second.
(4) Input an address followed by the displacement index and it will output the address to which a relative jump will be made.
(5) Performs addition, subtraction, multiplication and division on hex. numbers.

Using The Program

The program was written for a TI59 programmable calculator, but it will fit into a TI58 if the memory is repartitioned. The number of data registers used will depend upon the size of the hex. number processed, but for the average micro. using four digit addresses up to ten registers are used. Access to a print/security cradle is not required and, as it stands, the program contains no print commands.

To use the program it must first be initialized by the key sequence RST, CMS, R/S. The display will now show 0 and the TI59 is ready to receive its first number. The entry and readout of decimal numbers follows normal calculator practice. Hexadecimal numbers are a little different; firstly the six numerals A to F are represented by their decimal equivalents ie A=10 . . . . F=15; secondly multidigit numbers are entered one digit at a time, starting at the most significant, spearated by R/S, thus C50 is entered by the key sequence 12, R/S, 5, R/S, 0, R/S. The output of a hex. number takes place in a similar fashion. After a hex. calculation the most significant figure is in the display and operation of R/S brings each successive digit into the display unit a "flashing 1" indicates that all the information has been taken out. After the display has been cleared the calculator is ready for the next calculation.

When using the arithmetic routine it is necessary to enter the following codes for the arithmetic functions :–

addition = 0
subtraction = 1
multiplication = 2
division = 3

All other functions are obtained by the use of the user Labels and their use should be clear from the accompanying flow chart.

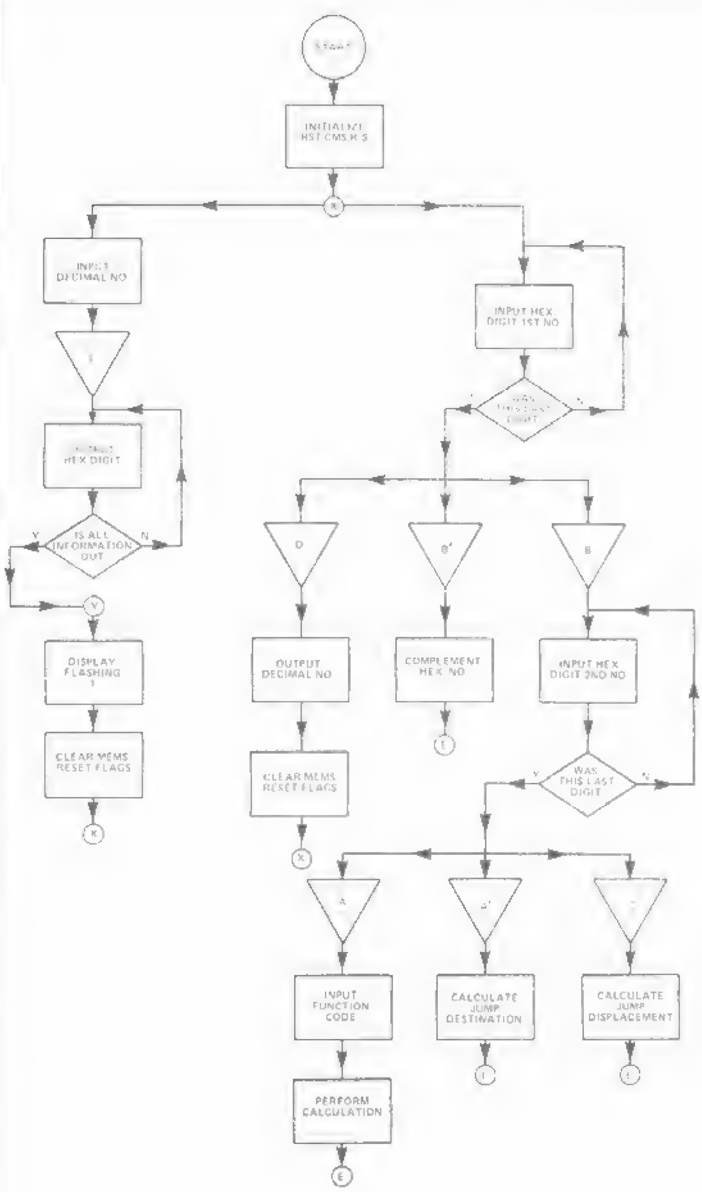
Sample Runs

	Keys Pressed			Display
	Program entered from magnetic cards.			
	RST, CMS, R/S			0
1)	15	R/S		15
	5	R/S	(Input F50)	5
	0	R/S		0
	B		(Signifies start of 2nd digit)	0
	15	R/S		15
	4	R/S	(Input F4)	4
	A		(Calculate jump destination)	0
	R/S			15
	R/S			4
	R/S			6
	R/S			1 (Flashing)

Thus a relative jump command with displacement of F4 (eg Z80 op.code 18 F4) located at 0F50 would cause a jump to location 0F46.

	CLR			0
2)	2	R/S		2
	11	R/S	(Input 2B)	11
	D		(Convert hex. to decimal)	43
3)	CLR			0
	13	R/S		13
	0	R/S	(Input D0B)	0
	11	R/S		11
	B		(Signifies start of 2nd digit)	11
	3	R/S		3
	10	R/S	(Input 3A)	10
	A		(Select arithmetic routine)	10
	0	R/S	(Select addition)	0
	R/S			13
	R/S			4
	R/S			5
	R/S			1(Flashing)
	CLR			0

Thus D0B + 3A = D45



PROGRAM LISTING

```
000  91  R/S
001  42  STO
002  00   00
003  87  IFF
004  00   00
005  65   ×
006  01   1
007  06   6
008  49  PRD
009  01   01
010  43  RCL
011  00   00
012  44  SUM
013  01   01
014  61  GTO
015  00   00
016  00   00
017  76  LBL
018  65   ×
019  01   1
020  06   6
021  49  PRD
022  02   02
023  43  RCL
024  00   00
025  44  SUM
026  02   02
027  61  GTO
028  00   00
029  00   00
030  76  LBL
031  13   C
032  01   1
033  94  +/-
034  32  X:T
035  43  RCL
036  02   02
037  75   -
038  43  RCL
039  01   01
040  95   =
041  42  STO
042  03   03
043  69  OP
044  10  E'
045  67   EQ
046  75   -
047  01   1
048  03   3
049  00   0
050  32  X:T
051  43  RCL
052  03   03
053  77   GE
054  24  CE
055  75   -
056  02   2
057  95   =
058  76  LBL
059  15   E
060  42  STO
061  04   04
062  61  GTO
063  43  RCL
064  76  LBL
065  75   -
066  01   1
067  02   2
068  06   6
069  94  +/-
070  32  X:T
071  43  RCL
072  03   03
073  22  INV
074  77   GE
075  24  CE
076  75   -
077  02   2
078  95   =
079  50  I×I
080  42  STO
081  04   04
082  86  STF
083  01   01
084  76  LBL
085  43  RCL
086  07   7
087  42  STO
088  06   06
089  01   1
090  42  STO
091  05   05
092  25  CLR
093  32  X:T
094  76  LBL
095  42  STO
096  43  RCL
097  04   4
098  55   ÷
099  01   1
100  06   6
101  95   =
102  42  STO
103  03   03
104  59  INT
105  42  STO
106  04   04
107  43  RCL
108  03   03
109  22  INV
110  59  INT
111  65   ×
112  01   1
113  06   6
114  95   =
115  72  ST*
116  06   06
117  69  OP
118  26   26
119  69  OP
120  25   25
121  43  RCL
122  04   04
123  22  INV
124  67   EQ
125  42  STO
126  22  INV
127  87  IFF
128  01   01
129  60  DEG
130  43  RCL
131  05   05
132  42  STO
133  03   03
```

```
134  43  RCL
135  06   06
136  42  STO
137  04   04
138  01   1
139  06   6
140  32  X:T
141  76  LBL
142  35  1/X
143  01   1
144  05   5
145  75   -
146  73  RC*
147  04   04
148  95   =
149  72  ST*
150  04   4
151  69  OP
152  34   34
153  97  DSZ
154  03   03
155  35  1/X
156  69  OP
157  24   24
158  43  RCL
159  05   05
160  42  STO
161  03   03
162  73  RC*
163  04   04
164  85   +
165  01   1
166  95   =
167  22  INV
168  77   GE
169  49  PRD
170  01   1
171  42  STO
172  15   15
173  25  CLR
174  76  LBL
175  49  PRD
176  72  ST*
177  04   04
178  69  OP
179  24   24
180  69  OP
181  33   33
182  76  LBL
183  48  EXC
184  73  RC*
185  04   04
186  85   +
187  43  RCL
188  15   15
189  95   =
190  77   GE
191  52  EE
192  72  ST*
193  04   04
194  25  CLR
195  42  STO
196  15   15
197  61  GTO
198  58  FIX
199  76  LBL
200  52  EE
201  25  CLR
202  72  ST*
203  04   04
204  76  LBL
205  58  FIX
206  69  OP
207  24   24
208  97  DSZ
209  03   03
210  48  EXC
211  76  LBL
212  60  DEG
213  73  RC*
214  06   06
215  91  R/S
216  69  OP
217  36   36
218  97  DSZ
219  05   05
220  60  DEG
221  76  LBL
222  24  CE
223  25  CLR
224  55   ÷
225  00   0
226  95   =
227  47  CMS
228  81  RST
229  76  LBL
230  12   B
231  86  STF
232  00   00
233  61  GTO
234  00   00
235  00   00
236  76  LBL
237  14   D
238  43  RCL
239  01   01
240  47  CMS
241  81  RST
242  76  LBL
243  17  B'
244  43  RCL
245  01   01
246  42  STO
247  04   04
248  86  STF
249  01   01
250  61  GTO
251  43  RCL
252  76  LBL
253  11   A
254  22  INV
255  87  IFF
256  00   00
257  24  CE
258  91  R/S
259  65   ×
260  05   5
261  85   +
262  02   2
263  07   7
264  02   2
265  95   =
266  42  STO
267  00   00
268  43  RCL
269  01   01
270  83  GO*
271  00   00
272  85   +
273  43  RCL
274  02   02
275  95   =
276  15   E
277  75   -
278  43  RCL
279  02   02
280  95   =
281  15   E
282  65   ×
283  43  RCL
284  02   02
285  95   =
286  15   E
287  55   ÷
288  43  RCL
289  02   02
290  95   =
291  15   E
292  76  LBL
293  16  A'
294  22  INV
295  87  IFF
296  00   00
297  24  CE
298  01   1
299  02   2
300  08   8
301  32  X:T
302  43  RCL
303  02   02
304  22  INV
305  77   GE
306  70  RAD
307  75   -
308  02   2
309  05   5
310  04   4
311  85   +
312  43  RCL
313  01   01
314  95   =
315  15   E
316  76  LBL
317  70  RAD
318  85   +
319  02   2
320  85   +
321  43  RCL
322  01   01
323  95   =
324  15   E
325  00   0
326  00   0
018  65   ×
031  13   C
059  15   E
065  75   -
085  43  RCL
095  42  STO
142  35  1/X
175  49  PRD
183  48  EXC
200  52  EE
205  58  FIX
212  60  DEG
222  24  CE
230  12   B
237  14   D
243  17  B'
253  11   A
293  16  A'
317  70  RAD
```

Stephen Draper.

BASIC PONTOON

The program given here will play pontoon against one opponent, the computer being banker. It is written in standard BASIC and uses no graphics; it should therefore be a fairly simple task to convert it to run on any BASIC using computer.

Program Notes

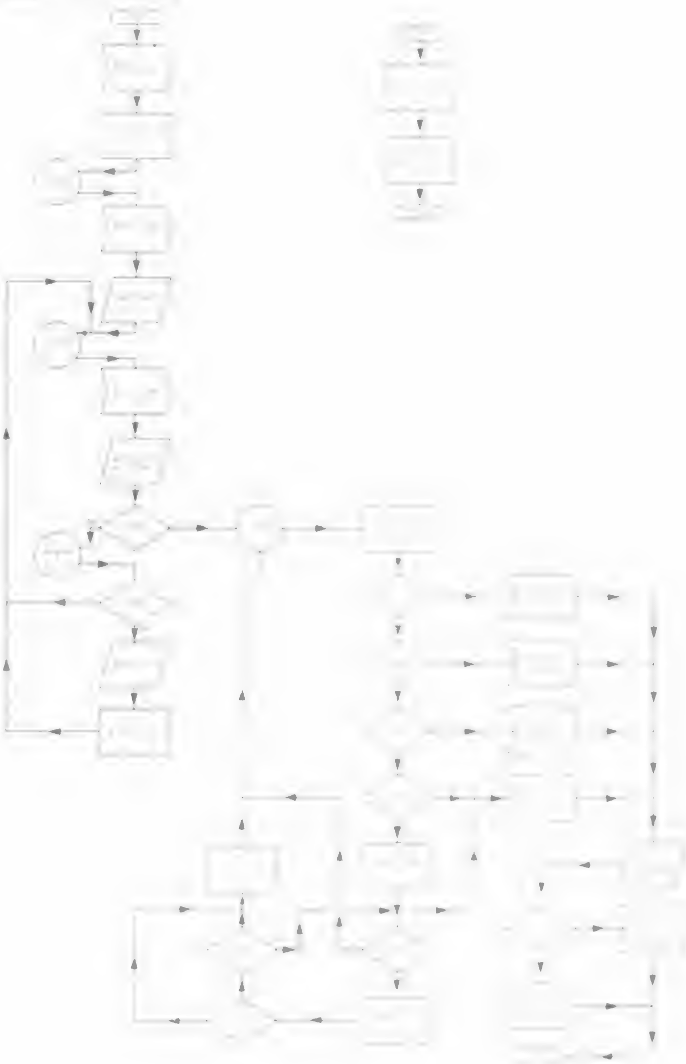
There are however a few points to note about playing the game :

1) If the computer's opponent has a pontoon he should tell the computer that his points value is 23.

2) Similarly if he has a five carder he should tell the computer that his points value is 22; for all other hands the points value is that of all the cards added together.

3) If the player is bust a value of 0 should be entered.

```
  0 DIM A(52)
  5 PRINT“PONTOON”
 10 LET A=1,B=1,C=1,N=0,T=0,Y=1
 15 INPUT“DO YOU WANT TO PLAY? Y OR N”E
 20 IF E=0 THEN 230
 25 LET A(A)=A
 30 IF A=52 THEN 45
 35 LET A=A+1
 40 GOTO 25
 45 R=RND(52)
 50 LET X=A(R),Z=A(B)
 55 LET A(R)=Z,A(B)=X
 60 IF B=52 THEN 75
 65 LET B=B+1
 70 GOTO 55
 75 GOSUB 300
 80 PRINT“YOUR CARD IS...”V
 85 INPUT“YOUR BID PLEASE”F
 90 GOSUB 300
 95 PRINT“YOUR CARD IS...”V
100 INPUT“STICK(1),TWIST(2),OR BUY(3),1,2 OR 3”H
105 IF H=1 THEN 140
110 GOSUB 300
115 IF H=2 THEN 90
125 INPUT“YOUR BID PLEASE”J
'30 LET F=F+J
135 GOTO 90
140 LET Z=C
145 GOSUB 300
150 LET T=T+V
155 LET N=N+1
160 IF(T=21)AND(N=2)THEN 250
165 IF T>21 THEN 210
170 IF N=5 THEN 235
175 IF T>16 THEN 270
180 FOR X=Z TO C
185 IF A(X)=1 THEN 200
190 NEXT X
195 GOTO 145
200 IF (T+10)>16 THEN 262
205 GOTO 145
210 PRINT“I AM BUST...”
215 PRINT“YOU HAVE WON“F”CREDITS”
220 INPUT“DO YOU WISH TO CONTINUE? Y OR N”W
225 IF W=1 THEN 10
230 STOP
235 PRINT“I HAVE A FIVE CARDER!”
240 LET T=22
245 GOTO 285
250 PRINT“PONTOON!!!...”
255 PRINT“...YOU HAVE LOST“F”CREDITS”
260 GOTO 220
262 IF (T+10)>21 THEN 145
265 LET T=T+10
267 GOTO 160
270 PRINT“STICK...I HAVE...WAIT FOR IT...”
275 FOR P=1 TO 30;NEXT P
280 PRINT“...”T
285 INPUT“WHAT DO YOU HAVE”Q
290 IF Q>T THEN 215
295 GOTO 255
300 IF A(C)<14 THEN 315
305 LET A(C)=A(C)-13
310 GOTO 300
315 IF A(C)>10 THEN A(C)=10
320 LET V=A(C)
325 LET C=C+1
330 RETURN
```



S.Hueber.

PINBALL

This program will emulate a pinball machine on a 4 or 8K PET. It should be noted that the 'Q' and 'S' characters in lines 193, 367 and 513 are cursor control symbols, cursor down and home respectively.

Program Notes
Line Nos.

9	Player starts off with one game.
10 – 100	Instructions.
110 – 196	Set up pin-table.
197 – 205	Give ball initial position and direction. Reset drop targets.
210 – 215	Put ball into play.
220 – 300	Process selected depending upon contents of next location in ball's path.
320 – 330	Bat control.
340 – 360	Calculates next location and tests for ball out of play.
362	Same ball again if no points scored.
363 – 365	Tests for final ball.
367 – 370	End of game messages.
449 – 530	Subroutines.
449 & 459	Limits of bat movement.
500	Prevents ball standing still!
511 – 512	Counts drop targets hit. If all hit extra ball awarded.
513 – 519	Prints score, tests for replays, prints replays.

```
9   CR=1
10  PRINT"DO YOU WANT INSTRUCTIONS?(Y OR N)"
20  GOSUB 520
30  IFA$="N" THEN 110
35  PRINT:PRINT
40  PRINT"3 BALLS PER GAME. PRESSING '1'
    MOVES"
50  PRINT"BAT 1 SPACE TO LEFT, '2' MOVES IT TO"
60  PRINT"RIGHT. BAT DETERMINES NEW
    DIRECTION OF"
70  PRINT"BALL ACCORDING TO WHERE ON BAT
    BALL"
80  PRINT"LANDS."
81  PRINT"COMPLETING DROP TARGETS SCORES"
82  PRINT"EXTRA BALL. MAXIMUM 1 EXTRA BALL"
83  PRINT"PER BALL IN PLAY."
84  PRINT" 1 REPLAY AWARDED WHEN 50 POINTS"
85  PRINT"SCORED. 1 REPLAY FOR EACH"
86  PRINT"ADDITIONAL SCORE OF 20 POINTS."
87  PRINT"TO GET EACH BALL INTO PLAY PRESS"
90  PRINT"ANY KEY."
100 PRINT:PRINT:PRINT"PRESS ANY KEY TO
    CONTINUE"
105 GOSUB 520
110 CR=CR-1
120 PRINT"     ";:FORN=32810TO32820:POKEN,100:
    NEXT:POKE32849,78:POKE32861,77
130 FORN=32888TO33408STEP40:POKEN,103:NEXT
140 FORN=32902TO33422STEP40:POKEN,101:NEXT
150 B=33415:POKEB-1,233:POKEB,160:POKEB+1,223
160 N=32809:POKEN,78:POKEN+39,78:POKEN+12,77:
    POKEN+53,77
170 X=33135
175 POKEX-123,15:POKEX-117,15
180 POKEX-2,15:POKEX+2,15
185 POKEX+78,15:POKEX+82,15
190 PRINTTAB(20);"BALL IN PLAY 0"
193 PRINT"Q";TAB(20);"CREDIT"
194 GOSUB518
195 S=0
196 N=1
197 IY=-1:IX=2:GOSUB490
198 P=32855:X1=7+IX:Y1=21
199 T=32895+IX
200 POKE32801,N+48
201 X=32852:FORY=XTOX+2
202 POKEY,90:NEXT:FORY=X+4TOX+6
203 POKEY,90:NEXT
204 E=0
205 S1=S
210 GOSUB520
215 POKEP,81
220 Q=PEEK(T)
230 IFQ=32ORG=96THENPOKEP,32:P=T:POKEP,81:
    X=X1:Y=Y1
235 IFQ=90THENGOSUB511
240 IFQ=103ORQ=101THENIX=-IX:IY=IYΛIY
250 IFQ=100THENIY=-IY
260 IFQ=15THENGOSUB513:GOSUB470
270 IFQ=77ORQ=78THENIX=-IX:IY=-IY
280 IFQ=233THENIX=-1:IY=1
290 IFQ=160THENIX=0:IY=1
300 IFQ=223THENIX=1:IY=1
310 FORD=1TO50:NEXT
320 GETD
330 ONDGOSUB449,459
340 X1=X+IX:Y1=Y+IY:T=33728+X1-40*Y1
350 IFT<33768THEN220
360 POKEP,32
362 IFS1=STHENN=N-1
363 N=N+1
365 IFN<4THEN197
367 PRINT"QQQQQQQQQQQQQQQQ"
368 IFCR=0THEN533
370 PRINT"PRESS 'R' FOR NEXT GAME"
380 GOSUB520
390 IFA$="R"THEN110
440 STOP
449 IFB=33411THENRETURN
450 POKEB+1,32:POKEB,223:POKEB-1,160:POKEB-2,
    223:B=B-1
452 RETURN
459 IFB=33419THENRETURN
460 POKEB-1,32:POKEB,233:POKEB+1,160:POKEB+2,
    223:B=B+1
463 RETURN
470 D=INT(RND(1)*3-1):IFD=IYTHEN470
480 IY=D
490 D=INT(RND(1)*3-1):IFD=IXTHEN490
500 IX=D:IFIX=0ANDIY=0THEN490
```

```
510 RETURN
511 POKET,32:E=E+1
512 IFE=6THENN=N-1
513 S=S+1:PRINT"S";S:IFS<50THENRETURN
514 IFS=50THEN517
515 IFINT((S-50)/20)=(S-50)/20THEN517
516 RETURN
517 CR=CR+1
518 IFCR<10THENPOKE32876,CR+48:RETURN
519 D=INT(CR/10):POKE32875,D+48:POKE32876,
    CR-D*10+48:RETURN
520 GETA$:IFA$=""THEN520
530 RETURN
533 PRINT"FOR ANOTHER GAME INSERT 10P COIN"
534 PRINT"(OR RUN THE PROGRAM AGAIN)"
540 END
```

Tony Lacy.

VARIABLE SAVER

TRS 80 owners may be interested in this subroutine which allows a small number of variables to be stored in a data statement during program execution. The program can then be 'CSAVED' and the stored data will be available for future use (no messing about with data tapes).

I use this method in programs such as personal accounts, storing previous best scores in games etc.

Program Note

The value of P is 17132 for the first character of the DATA statement.

```
1 DATA000000000000000000000000000000000000
  0000000000000000000000
2 REM THE DATA STATEMENT IS PUT AT THE
  START
3 REM THIS AVOIDS HAVING TO ALTER THE POKE
4 REM ADDRESSES IF THE PROGRAM IS MODIFIED
5 REM IT MUST CONTAIN MORE PADDING THAN
  THE
6 REM MAXIMUM NUMBER OF CHARACTERS TO
  BE
7 REM STORED (PLUS COMMAS AND QUOTES)
8 REM SOME DUMMY VARIABLES CAN BE
  INSERTED
9 REM IF THE READ STATEMENT WILL BE
10 REM ENCOUNTERED BEFORE THE STORAGE
11 REM SUBROUTINE
12 REM
13 REM
14 REM MAIN PROGRAM CAN BE HERE
16 REM
17 REM P IS THE POKE ADDRESS OF THE
18 REM START OF THE DATA STEMENT
19 REM N IS THE VALUE OF THE VARIABLE
20 REM TO BE STORED
21 REM N AND P ARE PROVIDED BY THE
22 REM CALLING PROGRAM
700 REM------------------
710 REM IN PROGRAM STORAGE ROUTINE
720 N$=STR$(N)
730 L=LEN(N$)
740 FOR P1=1 TO L
750 N1$=MID$(N$,P1,1)
755 IFN1$=' 'THEN 770
760 POKE(P1+P),ASC(N1$)
770 NEXT P1
780 POKE(P1+P),44
785 P=P1+P
790 RETURN
```

J.R. Keneally.

24 HOUR CLOCK

With reference to my article on the NASCOM scheduler in the October issue, may I correct a small error in the program suggested for testing the scheduler. This should have read :–

```
>ME00
0E00> 3A 98 0A 3C 32 98 0A C9
etc.
```

I have also found that the following code will always reset the PIO without the need to power-down. If desired, a similar sequence can be patched into the initialisation code of the scheduler.

The following code should be executed at address (0F00)

```
>MF00
0F00> CD 0C 0F 3E 03 D3 06
0F07> D3 07 C3 86 02 ED 4D
>
```

Timer Program

This program can be run under control of the NASCOM scheduler described in the October issue of CT, and will allow you to see how much time you are wasting on your computer. The program displays time in hours, minutes and seconds on the top line of the TV. It could be adapted to provide an alarm, or switch devices on and off at specific times of the day (or run a program at a specific time).

For accuracy, the counter/divider chain used to drive the TV timing is employed to provide an accurate clock for toggling the PIO, by connecting pin 13 of IC4 to pin 1 of SKA. This results in the basic timing interval of 16,384 micro-secs, if the 16 MHz crystal is accurate. Since the program is installed to run every 16 clock intervals, the time counters are advanced by 262,144 micro-secs for each activation.

The program holds time in micro-secs, milli-secs, secs., minutes and hours, using a 24-hour system. The first two of these quantities is held in binary form as a number between 0 and (999). The other three are held in BCD (Binary Coded Decimal) form, using one 8-bit byte for each quantity.

Program Use

To instal the program, use the sequence :–

```
>MC72
0C72>10 10 4 20 0F
>
```

The NASCOM monitor can be used almost normally at the same time as this program. The only command which should not be used is the Single-Step command. To set the time, locations 0FAA, 0FAB, 0FAC can be preset with the correct time in seconds, minutes and hours respectively. There is no need to stop the clock to make the change. Thus, the time can be set 09:29 and 30 seconds by using :–

```
>M0FAA
0FAA>30 29 09
>
```

The clock accuracy can be adjusted by putting 16-bit numbers into locations (0F24, 0F25) and (0F3C, 0F3D). The first locations define the number of whole micro-secs. which elapse between entries of the program, millisecs. elapsed. For example, if the time between program activations is 262,144 micro-secs., this is interpreted as 262 millisecs. plus 144 microsecs. Thus the above locations as set as shown in the program coding.

```
"24-HOUR CLOCK DISPLAY PROGRAM
"USE WITH NASCOM SCHEDULER.

0F20  2A 4C 0F                  MOV HL, (MICSEC)
0F23  11 90 00                  MOV DE, 144
                                "DE IS NO. OF MIC-SECS.
0F26  01 01 00                  MOV BC, 1
0F29  19                        ADD HL, DE
0F2A  11 E8 03                  MOV DE, 1000
0F2D  AF ED 52                  XOR A; SUB,C HL,DE
                                "CHECK IF MORE THAN 1000
0F30  30 02                     JR,NC L1
0F32  4F 19                     MOV C,A; ADD HL,DE
0F34  22 4C 0F          L1    : MOV (MICSEC),HL
0F37  2A 4E 0F                  MOV HL, (MILSEC)
0F3A  09                        ADD HL,BC
0F3B  11 06 01                  MOV DE, 262
                                "DE IS NO. OF MILLI-SECS.
0F3E  19                        ADD HL,DE
0F3F  11 E8 03                  MOV DE, 1000
0F42  A7 ED 52                  AND A; SUB,C HL,DE
0F45  30 09                     JR,NC L2
0F47  19                        ADD HL,DE
0F48  22 4E 0F                  MOV (MILSEC),HL
0F4B  C9                        RET
"DEFINE SOME DATA AREAS
0F4C  00 00             MICSEC  : AD 0
0F4E  00 00             MILSEC  : AD 0
"CONTINUE PROGRAM CODE
0F50  22 4E 0F          L2    : MOV (MILSEC),HL
0F53  21 AA 0F                  MOV HL,TSEC
0F56  7E A7                     MOV A, (HL); AND A
0F58  C6 01 27                  ADD 1; ADJ
0F5B  77                        MOV (HL),A
0F5C  FE 60 28 12               CP 96; JR,Z X1
0F60  21 AC 0F          X3    : MOV HL,THRS
0F63  11 F0 0B                  MOV DE, *5760
                                "DE IS TV ADDRESS
0F66  CD 92 0F                  CALL DISP
0F69  2B                        DEC HL
0F6A  CD 92 0F                  CALL DISP
0F6D  2B                        DEC HL
0F6E  CD 92 0F                  CALL DISP
0F71  C9                        RET
0F72  36 00 23          X1    : MOV (HL),0; INC HL
0F75  7E A7                     MOV A, (HL); AND A
0F77  C6 01 27                  ADD 1; ADJ
0F7A  77                        MOV (HL),A
0F7B  FE 60 28 02               CP 96; JR,Z X2
0F7F  18 DF                     JR X3
0F81  36 00 23          X2    : MOV (HL),0; INC HL
0F84  7E A7                     MOV A, (HL); AND A
0F86  C6 01 27                  ADD 1; ADJ
0F89  77                        MOV (HL), A
0F8A  FE 24 20 D2               CP 36; JR,NZ X3
0F8E  36 00                     MOV (HL), 0
0F90  18 CE                     JR X3
0F92  7E                        MOV A,(HL)
                                "DISPLAY TIME ROUTINE
0F93  E6 F0                     AND *360
0F95  CB 07 CB 07               SHFT,LCER A!&
0F99  CB 07 CB 07               SHFT,LCER A!&
0F9D  C6 30 12                  ADD 48; MOV (DE),A
0FA0  13 7E                     INC DE; MOV A,(HL)
0FA2  E6 0F C6 30               AND 15; ADD 48
0FA6  12 13 13 C9               MOV (DE),A;INC DE!&; RET
0FAA  00                TSEC  : DT 0 "SECONDS
0FAB  00                        DT 0 "MINUTES
0FAC  00                THRS  : DT 0 "HOURS
0FAD                            END
```

R.E.C. White.

NUMBER GAME

The program is written for Triton in V5.1 BASIC. The game is a simple idea, but one which is quite challenging to do. The computer prints a sequence of seven numbers which stay on the screen for a short while then disappear, then you have to type in the sequence of numbers that was displayed.

The computer will print a maximum of ten sequences of numbers providing you get each one correct, but it will print them on the screen for a shorter time so that when it gets to the tenth sequence the numbers are on the screen for a very short time. If you answer incorrectly the computer prints the correct answer followed by your score, then a new game is invited.

Program Modification

If the user has level 4.1 BASIC the following lines to be changed:–

```
 35 VDU 0, 12 FOR I = 1 TO 250; NEXT I
130 VDU 0, 12 FOR I = 1 TO 250; NEXT I
275 VDU 0, 12 FOR I = 1 TO 250; NEXT I
```

The game runs in 2K of memory.

```
  1  PRINT "NUMBER GAME"
  2  PRINT "-------      "
  3  PRINT "THIS IS A GAME WHERE THE COMPUTER PRINTS"
  4  PRINT "A ROW OF SEVEN NUMBERS : THEN AFTER A"
  5  PRINT "SHORT WHILE THE NUMBERS DISAPPEAR AND"
  6  PRINT "YOU HAVE TO TYPE IN THE SEQUENCE THAT WAS"
  7  PRINT "DISPLAYED. THE AMOUNT OF TIME THE NUMBERS"
  8  PRINT "ARE ON THE SCREEN GETS SHORTER AS YOU ANSWER"
  9  PRINT "CORRECTLY. THE MAXIMUM SCORE IS 10"
 10  FOR I=1 TO 6000; NEXT I
 11  LET R=1
 20  LET Z=0
 30  LET S=7000
 35  CALL 8
 40  FOR I=1 TO 7
 50  LET A=RND (9)
 60  LET @(I)=A
 70  NEXT I
 90  PRINT "SEQUENCE NUMBER",R
100  PRINT
110  PRINT #2, @(1), @(2), @(3), @(4), @(5), @(6), @(7)
```

```
120 FOR I=1 TO S; NEXT I
130 CALL 8
140 PRINT "NOW ENTER YOUR ANSWER ONE NUMBER AT A"
150 PRINT "TIME PRESSING RETURN AFTER EACH ONE"
160 PRINT
170 INPUT "1ST NUMBER"A; IF A#@(1) GOTO 275
180 INPUT "2ND NUMBER"B; IF B#@(2) GOTO 275
190 INPUT "3RD NUMBER"C; IF C#@(3) GOTO 275
200 INPUT "4TH NUMBER"D; IF D#@(4) GOTO 275
210 INPUT "5TH NUMBER"E; IF E#@(5) GOTO 275
220 INPUT "6TH NUMBER"F; IF F#@(6) GOTO 275
230 INPUT "7TH NUMBER"G; IF G#@(7) GOTO 275
240 PRINT
250 PRINT "CORRECT SEQUENCE";LET Z=Z+1, R=R+1
253 FOR I=1 TO 1000; NEXT I
255 IF R > 10 GOTO 310
260 LET S=S-500
270 GOTO 35
275 CALL 8
280 PRINT "INCORRECT NUMBER. THE SEQUENCE WAS:"
290 PRINT #2, @(1), @(2), @(3), @(4), @(5), @(6), @(7)
300 PRINT
310 PRINT "YOUR TOTAL WAS",Z
315 LET Y=1, N=0
320 INPUT "ANOTHER GAME &Y OR N)?"X
330 IF X=1 GOTO 11
340 STOP
```

Mark Williams.

MK14 AMBUSH

This program was written for the MK14 and is a space ambush program based on the Ambush project in the April issue of the ETI. The attacks come from two directions, either from the left or the right of the display. Your ship is at the centre of the display and you must press 1 or 3 to ward off the attacks. You have to press 1 if the attack is from the left and 3 if the attack is from the right. You will be attacked by twenty 'Yappanies' space ships and if you can successfully destroy them all you will have survived. (This number can easily be changed.)

You have limited energy, so you must keep the keys pressed for as short a time as possible. If you run out of energy, it will be indicated by the shape of your ship changing to three horizontal lines, and you will then be destroyed by the next attack.

The delay between each attack is random, as is the direction. The amount of energy and the speed of each attack can easily be changed to suit the user.

To play again, press 0. The program does not use the monitor so could be used with any SC/MP machine.

```
0F12   DIRECTION          0F15   ENERGY
0F13   DELAY              0F16   DISPLAY SHAPE
0F14   NUMBER LEFT        0F17   COUNT

0F12 01          DIRECTION:
0F13 25          DELAY:
0F14 14          NO. LEFT:
0F15 25          ENERGY:
0F16 3F          DISPLAY SHAPE:
0F17 00          COUNT:
0F18 C4 0F 36    BEGIN:   LDI X'0F XPAH 2   ;POINTER 2    0F00
0F1B C4 00 32             LDI X'00 XPAL 2
0F1E C4 0D 35             LDI X'0D XPAH 1   ;POINTER 1    0D00
0F21 C4 00 31             LDI X'00 XPAH 1   ;DISPLAY ADDRESS
0F24 BA 13                DLD X'13 2        ;DELAY UP?
0F26 9A 4D                J2 CONT           ;IF SO, ATTACK
0F28 C2 16                LD X'16 2         ;DISPLAY SHAPE
0F2A C9 04                ST X'04 1         ;OF YOUR SHIP
0F2C 8F 50                DLY               ;WAIT
0F2E C4 00                LDI X'00
0F30 C9 01                ST X'01 1
0F32 C9 03                ST X'03 1
0F34 C1 01                LD X'01 1         ;KEY 1 PRESSED?
0F36 E4 FF                XRI X'FF          ;IF SO DECREMENT
0F38 9C 06                JNZ X'06          ;FUEL
0F3A C1 03                LD X'03 1         ;KEY 3 PRESSED?
0F3C E4 FF                XRI X'FF          ;IF SO DECREMENT
0F3E 98 0C                JZ X'0C           ;FUEL
0F40 C2 15                LD X'15 2
0F42 98 04                JZ X'04           ;FUEL GONE?
0F44 BA 15                DLD X'15 2        ;IF SO STORE
0F46 90 04                JMP X'04          ;DIFFERENT SHAPE
0F48 C4 49                LDI X'49          ;AT CENTRE OF
0F4A CA 16                ST X'16 2         ;DISPLAY
0F4C 92 17                JMP BEGIN         ;GO TO START
0F4E C2 12       CONT:    LD X'12 2         ;DETERMINE ATTACK
0F50 9C 09                JNZ X'09          ;DIRECTION
0F52 C4 01                LDI X'01
0F54 CA 17                ST X'17 2
0F56 C4 00                LDI X'00
0F58 01                   XAE
0F59 90 07                JMP X'07
0F8B C4 FF                LDI X'CC
0F50 CA 17                ST X'17 2
0F5F C4 09                LDI X'09
0F61 01                   XAE
0F62 C4 40                LDI X'40          ;DISPLAY YOUR SHIP
0F64 CA 12                ST X'12 2
0F66 C2 16                LD X'16 2
0F68 C9 03                ST X'03 1
0F6A 8F 01                DLY
0F6C C4 41                LDI X'41          ;DISPLAY ATTACKER
0F6E C9 80                ST E(1)
0F70 8F 01                DLY
0F72 C4 00                LDI X'00
0F74 C9 01                ST X'01 1
0F76 C9 03                ST X'03 1
0F28 C2 17                LD X'17 2         ;WHICH KEY SHOULD
                                            BE PRESSED
0F7A E4 FF                XRI X'FF
0F7C 9C 08                JNZ X'08
0F7E C1 01                LD X'01 1         ;KEY PRESSED (1)
0F80 E4 FF                XRI X'FF
0F82 9C 1B                JNZ X'1B          ;IF SO TO 'HIT' PART
                                            OF PROGRAM
0F84 90 06                JMP X'06
0F86 C1 03                LD X'03 1         ;KEY PRESSED (3)
0F88 E4 FF                XRI X'FF
0F8A 9C 13                JNZ X'13          ;IF SO TO 'HIT' PART
                                            OF PROGRAM
0F8C BA12                 DLD X'12 2
0F8E 9E 65                JNZ X'65 2        ;DELAY UP?
0F90 C2 17                LD X'17 2         ;IF SO BRING ATTACKER
0F92 02                   CCL               ;1 CLOSER TO CENTRE
0F93 70                   CAE
0F94 01                   XAE
0F95 40                   LDE
0F96 E4 04                XRI X'04          HIT YOUR SHIP?
0F98 9E 61                JNZ X'61 2        ;IF NOT KEEP GOING
0F9A C4 E4                LDI X'E4
0F9C 32                   XPAL 2            ;IF SO DISPLAY "HIT"
0F9D 90 1A                JMP X'1A
0F9F C2 15                LD X'15 2         ;ENERGY LEFT?
0FA1 98 E9                JZ X'E9           ;IF SO WORK OUT DELAY
0FA3 C2 12                LD X'12 2         ;AND DIRECTION FROM
0FA5 D4 3F                ANI X'3F          ;RANDOM NUMBER
0FA7 02                   CCL               ;AND GO TO START
0FA8 F4 0F                ADI X'0F
0FAA CA 13                ST X'13 2
0FAC C2 12                LD X'12 2
0FAE D4 01                ANI X'01
0FB0 CA 12                ST X'12 2
0FB2 BA 14                DLD X'14 2        ;ANY ATTACKERS LEFT
0FB4 9E 17                JNZ X'17 2        ;IF SO START
0FB6 C4 ED                LDI X'ED          ;IF NOT DISPLAY
                                            SURVIVED
0FB8 32                   XPAL 2
0FB9 C4 09                LDI X'09
0FBB 01                   XAE
0FBC C2 80                LD E(2)           ;DISPLAY MESSAGE
0FBE C9 80                ST E(1)
0FC0 8F 01                DLY
0FC2 40                   LDE
0FC3 98 06                JZ X'06
0FC5 C4 FF                LDI X'FF
0FC7 02                   CCL
0FC8 70                   CAE
0FC9 90 F0                JMP X'F0
0FCB A9 00                ILD X'00 1        ;0 PRESSED?
0FCD 98 EA                JZ X'EA           ;IF SO, NEW GAME
0FCF C4 00                LDI X'00
0FD1 32                   XPAL 2
0FD3 C4 25                LDI X'25
0FD4 CA 13                ST X'13 2
0FD6 CA 12                ST X'12 2
0FD8 CA 15                ST X'15 2
0FDA C4 14                LDI X'14
0FDC CA 14                ST X'14 2
0FDE C4 3F                LDI X'3F
0FE0 CA 16                ST X'16 2
0FE2 92 17                JMP X'17
     00 00                END
DATA
0FE4    000,000,000,070,006,076,000,000,000         "HIT"
0FE     05E,079,03E,006,03E,031,03E,060,000       = "SURVIVED"
NOTE - TO INCREASE ENERGY, INCREASE BYTE AT 0FD3
       (ALSO INCREASES DELAY OF FIRST ATTACK)
       TO INCREASE NUMBER OF ATTACKERS, INCREASE BYTE AT 0FDB
```

Andrew Lack

TRITON CASSETTE CHECK

The 4.1 and 5.1 monitors on Triton do not provide any error checking facilities for the cassette interface. However, the UART (AY–5–1013) does provide error checking in-hardware, and the error flags can be accessed via port 01. The program below is intended to be loaded by hand into Triton's low RAM (1500 for example). The user can then verify any recording by using the program. The program is best used by checking recordings *before* switching off and losing the contents of the RAM!

Program Listing:

```
CD          START:    CALL TAPON          ;START CASSETTE DRIVE
DB 01       LOOP 1:   IN 01               ;READ STATUS BYTE
47                    MOV B,A             ;TEMPORARY STORE IN B
E6 01                 ANI 01              ;BIT 0 SET (DAV) ?
CA                    J2 LOOP             ;JMP IF NO
DB 04                 IN 04               ;RESET DAV FLAG
78                    MOV A,B             ;RESTORE A
E6 02                 ANI 02              ;BIT 1 SET (PE) ?
CA                    J2 CONT 1           ;JMP IF NO
11                    LXI D, STRING 1     ;STRING START ADDR.
CD 2B 00              CALL PSTRING        ;PRINT STRING
78          CONT 1:   MOV A,B             ;RESTORE A
E6 04                 ANI 04              ;BIT 2 SET (FE) ?
CA                    J2 CONT 2           ;JMP IF NO
11                    LXI D, STRING 2     ;STRING START ADDR.
CD 2B 00              CALL PSTRING        ;PRINT STRING
78          CONT 2:   MOV A,B             ;RESTORE A
E6 08                 ANI 08              ;BIT 3 SET (OR) ?
CA                    J2 LOOP 1           ;JMP IF NO
11                    LXI D, STRING 3     ;STRING START ADDR.
CD 2B 00              CALL PSTRING        ;PRINT STRING
C3                    JMP LOOP 1          ;NEXT BYTE
50 45 04    STRING1:  "PE" EOT            ;PARITY ERROR
46 45 04    STRING2:  "FE" EOT            ;FRAMING ERROR
4F 52 04    STRING3:  "OR" EOT            ;OVER RUN
```

END OF LISTING.

Notes :

1) The address of TAPON is :

   4.1 : 0327
   5.1 : 03F6

2) No addresses are given so that individual users may locate the routine where they wish.

D.C. Mower.

INTAB MOD

This program is intended as an extension to the excellent Intelligent Tabulator program published earlier in CT. The modification allows the start point to be entered from the keyboard without having to use the M command. This makes for a more flexible method when checking through programs in a random fashion.

```
0D20 EF 53 54          RST 40   S     T
  23 41 52 54            A      R     T      PRINT
  26 20 41 44                   A     D      START ADDRESS ?
  29 44 52 45            D      R     E      USING RST 40
  2C 53 53 20            S      S
  2F 3F 20 00            ?            E.O.T.
  32 21 52 0C   START    LD     HL    0C52   POINT TO HI ADDRESS STORE
  35 CD 3E 0D            CALL   INCH         CALL INPUT FOR HI BYTE
  38 CD 3E 0D            CALL   INCH         CALL INPUT FOR LO BYTE
  3B C3 50 0C            JP     0C50
  3E 06 02      INCH     LD     B     02H    DIGIT COUNT FOR INPUT
  40 CD 69 00   KEY      CALL   KBD          SCAN KDB FOR INPUT
  43 30 FB               JR     NC    KEY    LOOP UNTIL INPUT
  45 CD 3B 01            CALL   CRT          DISPLAY INPUT
  48 FE 39               CP     39H          TEST FOR ASCII NUMBER/LETTER
  4A 38 02               JR     C     NUMB   JUMP IF NUMBER,DON'T IF LETTER
  4C D6 07               SUB    07           REMOVE 37 IF LETTER
  4E D6 30      NUMB     SUB    30           REMOVE 30 IF NUMBER
  50 ED 6F               RLD                 STORE IN 4 LSB OF(HL)
  52 10 EC               DJNZ   KEY          LOOP IF FIRST DIGIT
  54 2B                  DEC    HL           POINT TO LO ADDRESS BYTE
  55 C9                  RET                 RETURN
```

NOTE : THE MODIFICATION ADDS 33H BYTES MAKING A TOTAL OF 105H BYTES
EXECUTE FROM 0D20 H.

Nigel Scales.

SPACE SHIP

Space Ship is a game written for the PET computer and is a dogfight simulation. The object of the game is to destroy your opponent (the computer) in the shortest time. There are ten targets and the score value of each decreases as time elapses. It should be noted that your controls work as though your craft is moving hence your target will appear to move in the opposite direction.

```
1   ? "[CLS] WOULD YOU LIKE THE INSTRUCTIONS"
2   INPUT A$ : IFA$ = "y" GOTO 5000
10  ? "[CLS] SPACE SHIP"
20  FOR X=1 TO 9999 :NEXT:? "[CLS]"
25  TI$ = "000000"
30  R=INT (RND(TI)*1000)+ 32768
40  GOSUB 2000
50  GET A$:IF A$: "  " GOTO 100
70  IF A$ = "M" GOTO 1000
75  IF A$ = "F" GOTO 500
80  A = VAL (A$)
90  ON A GOTO 150,200,300,400,100,600,700,800,900
100 D=RND(TI)
101 IF TI$ > = "000030" THEN TI$ = "000029"
102 L=30–VAL (RIGHT$(TI$,2))
103 ? "KILL RATIO";"[HOME] [SPACE] [SPACE]
    [HOME] ";L
105 GOSUB 3105
110 IF D < .125 THEN R=R+40
112 IF D > .125 AND D < .25 THEN R=R–39
114 IF D > .25 AND D < .375 THEN R=R–41
116 IF D > .375 AND D < .5 THEN R=R+1
118 IF D > .5 AND D < .625 THEN R=R+39
120 IF D > .625 AND D < .75 THEN R=R+41
122 IF D > .75 AND D < .875 THEN R=R–40
124 IF D > .875 THEN R=R–1
125 IF R > 33768 THEN R=R–40
126 IF R < 32768 THEN R=R+40
128 GOSUB 2000
```

```
130  GOTO 50
150  GOSUB 3105
200  GOSUB 3105
300  GOSUB 3105
400  GOSUB 3105
500  LA=33736 : LB=33760
510  FOR X=1 TO 12
515  POKE LA,78:POKE LB,77
520  LA=LA-39 : LB=LB-41
525  NEXT X
527  Z=33268
530  IF R=Z GOTO 560
540  FOR X=1 TO 13
541  POKE LA,32 : POKE LB,32
543  LA=LA+39 : LB=LB+41
545  NEXT X
550  GOTO 50
560  POKE Z,42:POKE Z-41,85:POKE Z-39,73:POKE
     Z+39,74: POKE Z+41,75
565  V=V+L:?"HOME";SPC(14);V;"POINTS OUT OF"
567  P=P+1
568  ? SPC(14);P;"SHIPS"
569  IF P<10 GOTO 20
570  ?"[CLS] TOTAL KILL RATIO";V:GOTO 1000
600  GOSUB 3105:R=R-1:GOSUB 2000:GOTO 50
700  GOSUB 3105:R=R+41:GOSUB 2000:GOTO 50
800  GOSUB 3105:R=R+40:GOSUB 2000:GOTO 50
900  GOSUB 3105:R=R+39:GOSUB 2000:GOTO 50
1000 ?"WOULD YOU LIKE ANOTHER GAME?"
1010 GET A$: IF A$= "  " GOTO 1010
1015 IF A$= "N" THEN END
1020 RUN 20
2000 POKE R,209:POKE R-41,77:POKE R+39,78:POKE
     R-39,78: POKE R+41,77
2100 RETURN
3105 POKE R,32:POKE R-41,32:POKE R+39,32:POKE
     R-39,32
3110 RETURN
5000 ?"YOU ARE AT THE HELM OF THE TERRAN"
5010 ?"SPACE FIGHTER XJFT 3202 ON THE TAIL OF
     AN ENEMY"
5020 ?"CRAFT WHICH YOU MUST TRY TO DESTROY"
5030 ?"YOU MAY ALTER YOUR COURSE,
     IN ORDER TO"
5040 ?"HOME IN ON YOUR QUARRY, USING KEYS 1
     TO 9 (NOT 5)"
5050 ?"PRESS THE 'F' KEY TO FIRE"
5060 ?"PRESS 'M' TO START AND STOP"
5070 GET A$: IF A$="  " OR A$<>"M" GOTO 5070
5080 RUN 20
```

Gary Hawkins.

STOPWATCH

For those of you who need a stop watch this program for the Nascom, or any Z80 based system, may suit your requirements. The program must be loaded into memory 0C50–0DAF as call and jump codes are within this area. Execution is from 0C50, pressing H allows the hours to be set with each depression of the key adding 1 to the counter. Minutes can be set in the same way using M. To reset the clock key R will reset and stop, S will start the counting again.

Display Format

Hours, minutes, seconds and hundredths are displayed in the centre of the screen.

```
0C50  3E 1E CD 3B 01   LD A, 1E CALL 'CRT'
0C55  21 9B 09         LD HL, 9B09
0C58  36 30 23         (HL), 30 INC HL
0C5B  36 30 23          "    "   "   "
0C5E  36 3A 23         (HL), 3A INC HL
0C61  36 30 23         (HL), 30 INC HL
0C64  36 30 23          "    "   "   "
0C67  36 3A 23         (HL), 3A INC HL
0C6A  36 30 23         (HL), 30 INC HL
0C6D  36 30 23          "    "   "   "
0C70  36 3A 23         (HL), 3A INC HL
0C73  36 30 23         (HL), 30 INC HL
0C76  36 30            (HL), 30
0C78  0E 14            LD C, 20H
0C7A  06 32 00         LD B, 50H NOP
0C7D  10 FD 0D         DJNZ -1 DEC C
0C80  20 F8            JRNZ -6
0C82  CD 69 00         CALL KB SKAN
0C85  FE 40            CP=@
0C87  CA EF 0C         JRZ OCEF
0C8A  FE 52            CP=R
0C8C  CA 0B 0D         JRZ 0D0B
0C8F  34 7E            INC (HL) A,(HL)
0C91  FE 3A            CP=:
0C93  28 03            JRZ +3
0C95  C3 78 0C         JR 0C78
0C98  2B 34 7E         DEC HL (HL) A,(HL)
0C9B  FE 3A            CP=:
0C9D  28 03            JRZ +3
0C9F  C3 75 0C         JR 0C75
0CA2  2B 2B            DEC HL DEC HL
0CA4  34 7E            (HL) A,(HL)
0CA6  FE 3A            CP=:
0CA8  28 03            JRZ +3
0CAA  C3 6F 0C         JR 0C6F
0CAD  2B 34 7E         DEC HL (HL) A,(HL)
0CB0  FE 36            CP=6
0CB2  28 03            JRZ +3
0CB4  C3 6C 0C         JR 0C6C
0CB7  2B 2B            DEC HL DEC HL
0CB9  34 7E            (HL) A,(HL)
0CBB  FE 3A            CP=:
0CBD  28 03            JRZ +3
0CBF  C3 66 0C         JR 0C66
0CC2  2B 34 7E         DEC HL (HL) A, (HL)
0CC5  FE 36            CP=6
0CC7  28 03            JRZ +3
0CC9  C3 63 0C         JR 0C63
0CCC  2B 2B            DEC HL DEC HL
0CCE  34 7E            (HL) A,(HL)
0CD0  FE 3A            CP=:
0CD2  20 05            JRNZ +5
0CD4  36 30            (HL),30
0CD6  2B 34            DEC HL (HL)
0CD8  23 2B            INC HL DEC HL
0CDA  7E               A,(HL)
0CDB  FE 32            CP=2
0CDD  20 09            JRNZ +9
0CDF  23 7E            INC HL A,(HL)
0CE1  FE 34            CP=4
```

0CE3	20	04		JRNZ +4
0CE5	C3	50	0C	JR 0C50
0CE8	23			INC HL
0CE9	C3	5D	0C	JR 0C5D
0CEC	00	00	00	NOPS (PADDING)
0CEF	CD	69	00	CALL KB SKAN
0CF2	FE	53		CP=S
0CF4	CA	78	0C	JRZ 0C78
0CF7	FE	52		CP=R
0CF9	CA	0B	0D	JRZ 0D0B
0CFC	FE	48		CP=H
0CFE	CA	36	0D	JRZ 0D36
0D01	FE	4D		CP=M
0D03	CA	76	0D	JRZ 0D76
0D06	18	E7		JR –23H
0D08	00	00	00	NOPS
0D0B	21	9B	09	LD HL, 9B09
0D0E	06	04		LD B, 04H
0D10	36	30	23	(HL), 30 INC HL
0D13	36	30	23	" " " "
0D16	36	3A	23	(HL), 3A INC HL
0D19	10	F5		DJNZ –9H
0D1B	2B	36	20	DEC HL (HL), 20
0D1E	2B			DEC HL
0D1F	CD	69	00	CALL KB SKAN
0D22	FE	53		CP=S
0D24	CA	78	0C	JRZ 0C78
0D27	FE	48		CP=H
0D29	CA	36	0D	JRZ 0D36
0D2C	FE	4D		CP=M
0D2E	CA	76	0D	JRZ 0D76
0D31	18	EC		JR –18H
0D33	00	00	00	NOPS
0D36	21	9C	09	LD HL, 9C09
0D39	34	7E		(HL) A,(HL)
0D3B	FE	3A		CP=:
0D3D	20	05		JRNZ +5
0D3F	36	30		(HL),30
0D41	2B	34		DEC HL (HL)
0D43	23	2B		INC HL DEC HL
0D45	7E			A,(HL)
0D46	FE	32		CP=2
0D48	20	0B		JRNZ +11H
0D4A	23	7E		INC HL A,(HL)
0D4C	FE	34		CP=4
0D4E	20	06		JRNZ +6
0D50	36	30		(HL),30
0D52	2B	36	30	DEC HL (HL),30
0D55	23			INC HL
0D56	CD	69	00	CALL KB SKAN
0D59	FE	48		CP=H
0D5B	28	DC		JRZ –34H
0D5D	FE	53		CP=S
0D5F	20	06		JRNZ +6
0D61	21	A5	09	LD HL, A509
0D64	C3	78	0C	JR 0C78
0D67	FE	4D		CP=M
0D69	CA	76	0D	JRZ 0D76
0D6C	FE	52		CP=R
0D6E	CA	0B	0D	JRZ 0D0B
0D71	18	E3		JR –27H
0D73	00	00	00	NOPS
0D76	21	9F	09	LD HL, 9F09
0D79	34	7E		(HL) A,(HL)
0D7B	FE	3A		CP=:
0D7D	20	10		JRNZ +16H
0D7F	36	30		(HL),30
0D81	2B	34	7E	DEC HL (HL) A,(HL)
0D84	FE	36		CP=6
0D86	20	06		JRNZ +6
0D88	36	30	23	(HL),30 INC HL
0D8B	36	30	2B	(HL),30 DEC HL
0D8E	23			INC HL
0D8F	CD	69	00	CALL KB SKAN
0D92	FE	4D		CP=M
0D94	28	E3		JRZ –27H
0D96	FE	53		CP=S
0D98	20	06		JRNZ +6
0D9A	21	A5	09	LD HL, A509
0D9D	C3	78	0C	JR 0C78
0DA0	FE	48		CP=H
0DA2	CA	36	0D	JRZ 0D36
0DA5	FE	52		CP=R
0DA7	CA	0B	0D	JRZ 0D0B
0DAA	18	E3		JR –27H
0DAC	00	00	00	NOPS

A.M. Scott.

SC/MP DICE

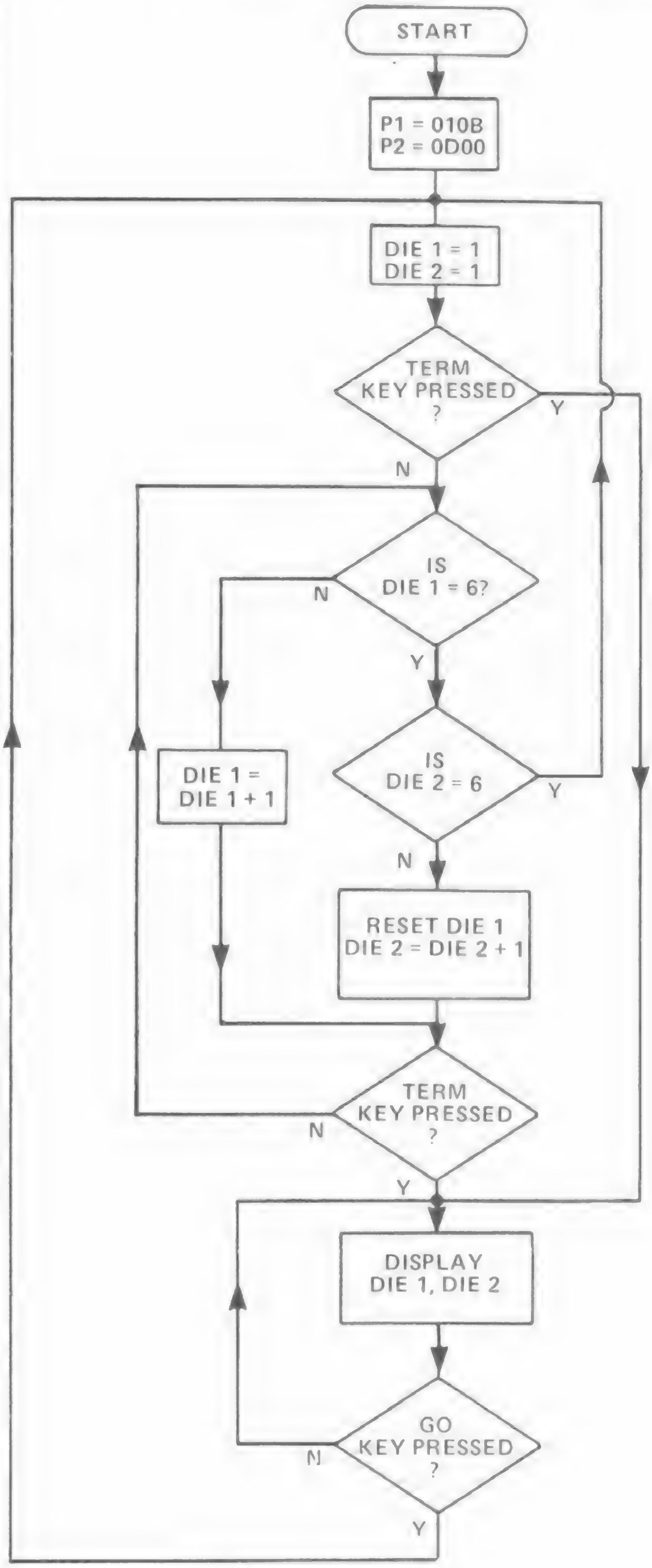
Many games require two dice to be thrown so this program was developed to display two independant dice throws on an MK14. The program starts at 0F20 but one location is used as a temporary store at 0F1F for die 1 while die 2 is held in the extension register. Go is pressed to roll the dice and Term throws them and displays the result.

Notes

Although the program was written on an MK14 it could easily be modified to use any SC/MP based system. For people wishing to modify, P1 holds the address of the seven segment display codes and P2 is used to set the display address and to access the keyboard.

0F1F			DIE 1	.=, +1	
20	C4	01		LDI	
22	35			XPAH (1)	P1 set to
23	C4	0B		LDI	segment codes
25	31			XPAL (1)	
26	C4	0D		LDI	P2 set to
28	36			XPAH (2)	display
29	C4	00		LDI	address
2B	32			XPAL (2)	
2C	C4	01	REPEAT	LDI	Start
2E	C8	F0		ST	Store DIE 1
30	01			XAE	Store DIE 2
31	AA	07		ILD (2)	Term Pressed
33	9C	19		JNZ	If so, show
35	C0	E9	TEST	LD	DIE 1 = 6?
37	E4	06		XRI	"
39	9C	0D		JNZ	1 per
3B	E4	06		LDE	DIE 2 = 6?
3C	E4	06		XRI	"
3E	98	EC		JZ	Repeat
40	C4	01		LDI	Reset DIE 1
42	C8	DC		ST	"
44	70			ADE	Increment

```
  45   01                XAE        DIE 2
  46   90 02             JMP        Press
  48   A8 D6   INCR      ILD        Incr. DIE 1
0F4A   AA 07   PRESS     ILD (2)    Term Pressed
  4C   98 E7             JZ         If not, Test
  4E   C0 D0   SHOW      LD         DIE 1
  50   C8 02             ST
  52   C1 00   AGAIN     LD (1)     Fetch Segments
  54   CA 02             ST (2)     Show DIE 1
  56   C1 80             LD (1)     Fetch Segments
  58   CA 05             ST (2)     Show DIE 2
  5A   C4 00             LDI        Clear
  5C   CA 05             ST (2)     Display
  5E   AA 02             ILD (2)    Go pressed
  60   9C CA             JNZ        If so, repeat
  62   90 EE             JMP        Again
```



M.G. Foster.

NASFORTE

This program uses the lower two rows of keys on the Nascom keyboard to simulate an electronic piano. The tone output is produced from pin 14 of SKT 1 (the keyboard socket) via a 100 nF capacitor to a suitable audio amplifier or high impedance headphones.

Which Key, What Note?

In an attempt to make the key locations similar to a conventional piano key 'C' is 'C' above middle 'C' and the range from Ab through one complete octave to C# is available, keys 'D', 'H' and '1/3' being unused.

Although the note length varies with pitch this has been found to be of no real disadvantage, the mean length of the note can be altered by modifying the values in DE register (0C64–65). The pitch value can be calculated from the formula

$$\text{Pitch Value (decimal)} = \frac{\text{Freq Clock (CPU)}}{\text{Freq Note 27}}$$

This gets you in the right 'ballpark' and slight adjustment should be made for optimum pitch, raising the value will lower the note.

One must convert the calculated decimal value to Hex and modify the appropriate location in the Note Table. The values published give the International scale, A=440 Hz, for a 2 MHz CPU clock.

Program Modifications

In order to dump your tunes onto tape the following changes are needed:–

```
0C63  7B           LDA E          Recover key
  64  CD 5D 00                    Call SRLOUT
  67  11 6F 01     LD DE 016F
etc.,
0C7C  18 D2
```

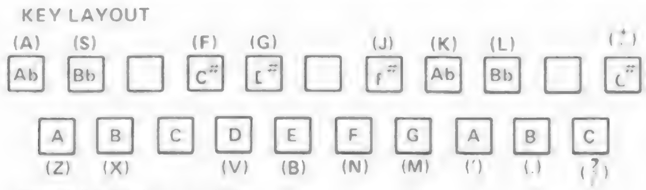
You can now start the tape transport and play the tune directly onto the tape. Note that when using the T4 monitor SRLOUT will also produce a screen listing.

Playing the Nasforte is somewhat different to a piano in that notes cannot be sustained, however runs can be achieved by pressing the next note key whilst still holding the previous one. It will certainly make a change from the cachophony of the children's stylophone!

Nas SYS Modifications

To run the program under Nas SYS monitors the following changes must be made:–

Relocate the program to 0C80, relocate the Note Tables to 0CB0, use R/START 'CF' in place of CHIN, location 0C83 becomes CF 00 00. Relative jumps can be left alone as the previous mods take care of this.



FREQUENCIES OF NOTES

Ab	415.3 Hz
A	440 Hz
Bb	466.2 Hz
B	493.9 Hz
C	523.2 Hz
C #	554.4 Hz
D	586.6 Hz
D #	622.2 Hz
E	659.2 Hz
F	698.4 Hz
F #	740.0 Hz
G	784.0 Hz
Ab	830.6 Hz
A	880.0 Hz
Bb	932.4 Hz
B	987.8 Hz
C	1046.4 Hz
C #	1109 Hz

```
0C50 21 80 0C     'NEXNOT'  LD HL NOTE TAB   HL HAS START OF NOTE TABLE
  53 CD 3E 00               CALL CHIN        KEY PRESSED?
  56 5F                     LD E   A         SAVE KEY
  57 7E           'FIND'    LD A (HL)        GET KEY CODE
  58 23                     INC HL           HL POINTS TO NOTE VALUE
  59 B7                     OR A             TEST FOR END OF TABLE
  5A 28 F4                  JRZ 'NEXNOT'     JUMP IF END (NOT FOUND)
  5C BB                     CP E             DOES TABLE VALUE - KEY?
  5D 28 03                  JRZ 'NOTE'       IF IT DOES 'PLAY IT'. IF NOT
  5F 23                     INC HL           TRY NEXT
0C60 18 F5                  JR 'FIND'
  62 4E           'NOTE'    LD C (HL)        GET NOTE VALUE
  63 11 6F 01               LD DE 016F       DE SET TO NOTE LENGTH
  66 41           'PITCH'   LD B   C         PASS NOTE TO PLAY
  67 3E 20                  LD A 20H         SET UP MASK FOR O/P PORT
  69 D3 00                  OUT 0   A        SET BIT 5 PORT '0'
  6B 10 FE                  DJNZ HERE        DELAY 1
  6D 41                     LD B   C
  6E AF                     XOR A            REMOVE MASK
  6F D3 00                  OUT 0   A        RESET BIT 5 PORT '0'
  71 10 FE                  DJNZ HERE        DELAY 2
  73 1B                     DEC DE           NOTE LENGTH COUNT   1
  74 7A                     LD A   D         TEST FOR END OF
  75 B7                     OR A             NOTE IF NOT
  76 20 EE                  JRNZ 'PITCH'     LOOP UNTIL END
0C78 18 D6                  JR 'NEXNOT'      GET NEXT NOTE
                        NOTE  TABLE DEF B
0C80 41 B6 5A AC  Ab A      EVEN ADDRESS     = ASCII KEY CODE
  84 53 A2 58 99  Bb B      ODD ADDRESS      = NOTE CODE
  88 43 90 46 88  C  C#
  8C 56 81 47 79  D  D#
  90 42 72 4E 6C  E  F
  94 4A 66 4D 60  F# G
  98 4B 5A 2C 55  Ab A
  9C 4C 50 2E 4B  Bb B
  A0 2F 47 3A 42  C  C#
0CA4 00 00 00 00            4 X NOPS
```

S.A.Bigg.

BLACK BOX

The "Black Box" game, which this program emulates, is a little like the old favourite "Mastermind". One player sets up a code which is broken by the second player but rather than using numbers or colours a large grid is used. Each square on the eight by eight grid has a number assigned to it and in the board version of the game one player marks where he is placing his 'atoms', there are either four or five by agreement. The object of the game is for the other player to discover where these are by deduction. To achieve this he sends in 'rays' at one of the thirty-two input locations. These rays may be absorbed by atoms in their direct path or reflected by atoms adjacent to their path, or a mixture of these two possibilities. The player who is attempting to break the code marks the information he receives from each attempt on a duplicate grid using coloured markers.

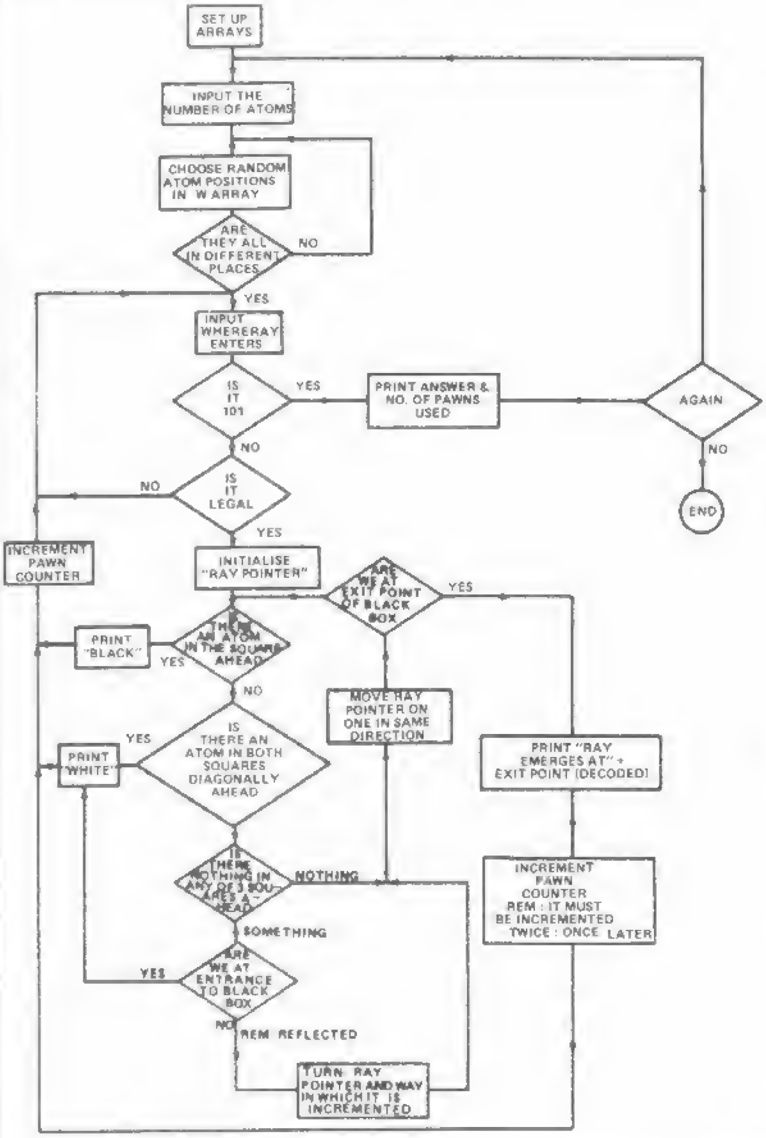
An absorbed ray is marked by a black pawn, a ray returning to its entry point is marked with a white pawn, as is one which enters a square directly adjacent to an atom. A ray which enters and leaves the box at different points has these points marked with a pair of pawns of the same colour. Some typical ray patterns are shown in the second diagram.

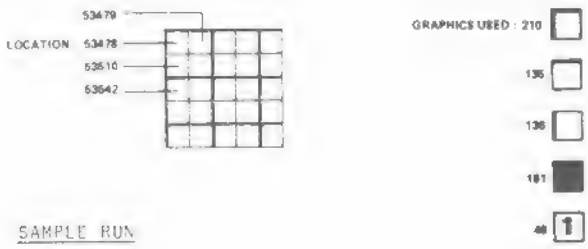
The Simulation Program

This program was written on the Ohio Superboard and uses about 3½K of RAM. It should be easy to adapt for any other system using Microsoft BASIC but screen locations and graphics will probably have to be altered. If it is loaded on the Superboard and you wish to save it you will have to set the line width to 71, as you would have to for any program with lines greater than 24 characters. There is a sample run given after the program listing showing how the computer takes the place of the first player and a flowchart to assist in conversion to other languages.

```
1   ?"BLACK BOX":?
4   INPUT "INPUT SEED";S
7   ?:?:?:?:?:?
10  DIM B(6),D(6),M(4),N(4),C(32)
11  DIM W(10,10)
30  DATA 0,1,-1,0,0,-1,1,0
40  DATA 2,3,4,5,6,7,8,9,10,10,10,10,10,10,10,10
50  DATA 10,9,8,7,6,5,4,3,2,1,1,1,1,1,1,1
80  FOR X=1 TO 4
90  READ M(X),N(X):NEXT
100 FOR X=1 TO 32
120 READ C(X):NEXT
140 INPUT"HOW MANY ATOMS";R
150 K=0
160 IF R<>4 AND R<>5 AND R<>6 THEN 140
170 FOR X=1 TO R
180 I=INT(RND(S)*8+2)
190 J=INT(RND(S)*8+2)
200 IF X=1 THEN 240
210 FOR Y=1 TO (X-1)
220 IF B(Y)=I AND D(Y)=J THEN 180
230 NEXT Y
240 B(X)=I:D(X)=J
250 NEXT X
270 FOR X=1 TO 10
280 FOR Y=1 TO 10
290 W(X,Y)=0
300 NEXT Y:NEXT X
320 FOR X=1  TO R
325 W(B(X),D(X))=1
330 NEXT
335 ?"TYPE 101 TO SEE ANSWER"
337 ?:?:?
340 INPUT "RAY ENTERS WHERE";P
345 Z=0
360 IF P=101 THEN 970
370 IF INT(ABS(P))<>P THEN 340
380 IF P<1 OR P>32 THEN 340
390 F=M(INT((P+7)/8))
400 G=N(INT((P+7)/8))
410 U=C(P)
420 IF P+24>32 THEN Q=P-8:GOTO 440
430 Q=P+24
440 V=C(Q)
450 A=U+F:B=V+G
460 E=0
470 IF W(A,B)=1 THEN 910
480 IF F=0 THEN 540
490 B=B+1
500 IF W(A,B)=1 THEN E=1
505 B=B-2
510 IF W(A,B)=1 OR E=1 THEN 580
530 GOTO 760
540 A=A+1
550 IF W(A,B)=1 THEN E=1
560 A=A-2
570 IF W(A,B)=0 AND E<>1 THEN 760
580 IF Z=0 THEN 930
590 IF F=0 THEN 620
600 IF W(A,B)*W(A,B+2)=1 THEN 930
610 GOTO 630
620 IF W(A,B)*W(A-2,B)=1 THEN 930
630 A=F+U:B=V+G
650 IF F=0 THEN 710
660 IF W(A,B+1)=1 THEN 690
670 F=0:G=1
680 GOTO 760
690 F=0:G=-1
700 GOTO 760
710 IF W(A+1,B)=1 THEN 740
720 F=1:G=0: GOTO 760
```

```
740  F=-1:G=0
760  U=U+F:V=V+G
770  IF U=1 OR U=10 OR V=1 OR V=10 THEN 820
780  Z=Z+1:GOTO 450
820  FOR X=1 TO 32
830  IF U<>C(X) THEN 870
840  IF X>8 THEN Q=X-8:GOTO 860
850  Q=X+24
860  IF V=C(Q) THEN 880
870  NEXT
880  ?"RAY RE-EMERGES AT";X
890  K=K+1:GOTO 940
910  ?"BLACK":GOTO 940
930  ?"WHITE"
940  K=K+1
950  GOTO 337
970  ?"THIS IS THE RIGHT ANSWER:":?
980  FOR X=1 TO 18:?:NEXT
990  FOR X=53478 TO 53926 STEP 64
994  FOR Y=0 TO 14 STEP 2
997  POKE Y+X,210:POKE Y+X+1,135
1003 NEXT Y
1004 POKE X+16,136
1005 FOR Y=32 TO 48 STEP 2
1007 POKE Y+X,136:NEXT Y:NEXT X
1013 FOR X=53990 TO 54005
1015 POKE X,135:NEXT X
1019 POKE 53476,49
1012 FOR X=2 TO 9
1023 FOR Y=2 TO 9
1024 IF W(X,Y)=0 THEN 1037
1025 T=(X-2)*64+(Y-2)*2+53478
1029 POKE T,161:POKE T+1,161:POKE T+32,161:POKE T+33,161
1037 NEXT Y: NEXT X
1080 ?"YOU USED ";K;"PAWNS"
1110 INPUT"AGAIN ?";A$
1111 IF LEFT$(A$,1)="Y" THEN 140
1119 IF LEFT$(A$,1)="N" THEN END
2000 GOTO 1110
```

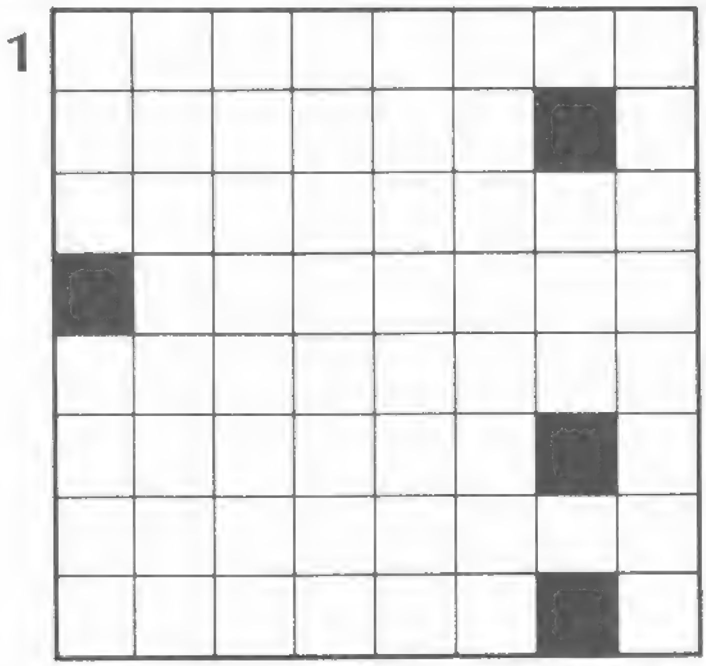




SAMPLE RUN

```
BLACK BOX

INPUT SEED ? 23.1

HOW MANY ATOMS ? -5.3
HOW MANY ATOMS ? 4
TYPE 101 TO SEE ANSWER

RAY ENTERS WHERE?  19
BLACK

RAY ENTERS WHERE ? 25
RAY RE-EMERGES AT 26

RAY ENTERS WHERE ? 28
RAY RE-EMERGES AT 13

RAY ENTERS WHERE ? 7
WHITE

RAY ENTERS WHERE ? 3
WHITE

RAY ENTERS WHERE ? 101
THIS IS THE RIGHT ANSWER:

YOU USED 7 PAWNS
AGAIN ? NO

OK
```

Below : a typical graphic display.



J. Allin.

TEXT EDIT

This text editing program is designed for an unexpanded Nascom 1. It is versatile and contains a number of useful features. 'Shift' function is provided by pressing the '@' key then the required character. The control characters used are:

- '@' followed by '<' moves cursor up one line
- '@' followed by '>' moves cursor down one line
- '@' followed by 'New Line' scrolls whole text down one line
- '@' followed by 'Back Space' clears screen

In addition,

- '@' followed by 'Space' and,
- '@' followed by '@'

can be used to jump into further subroutines (eg, dump text onto teletype).

Text Alteration

Moving the cursor up or down does not erase the text (only 'BS' and 'Space' keys will), making it simpler to correct errors in the middle of a page.

The program will also enable text to be written into the unscrolled top line (by moving the cursor down).

To execute, key E D00, followed by 'NL'

```
D00   3E 1E       LD A, 1E          Clear screen
D02   CD 3B 01    CALL CRT
D05   16 20       LD D, 20          load 'space' (for scroll down S.R.)
D07   CD 69 00    CALL KBD          scans keyboard
D0A   30 FB       JR NC -3
D0C   FE 40       CP 40             look for '[illegible]'
D0E   CA 27 0D    JP Z D27          jump 'shift' S.R. if [illegible]
D11   00 00 00    NOP
D14   00 00
D16   FE 3F       CP 3F             look for '?'
D18   F2 20 0D    JP P D20          jump if greater than '?'
D1B   C3 64 0D    JP D64            jump CRT S.R.
D1E   00 00       NOP
D20   C6 20       ADD 20            shift characters 41 to 4A
D22   C3 64 0D    JP D64
D25   00 00       NOP
D27   CD 69 00    CALL KBD
D2A   30 FB       JR NC -3          looks for character to be shifted
D2C   FE 2C       CP 2C             look for '<'
D2E   CA 90 0D    JP Z D90          jump if '<' to cursor S.R.
D31   FE 2E       CP 2E             look for '[illegible]'
D33   CA 90 0D    JP Z D90
D36   FE 2D       CP 2D             look for '[illegible]'
D38   CA 43 0D    JP Z D43
D3B   FE 2F       CP 2F             look for '[illegible]'
D3D   CA 43 0D    JP Z D43
D40   C3 48 0D    JP D48
D43   C6 10       ADD 10            shift characters ' -' & '[illegible]'
D45   C3 64 0D    JP D64
D48   FE 3C       CP 3C             look for '<'
D4A   F2 54 0D    JP P D54
D4D   D6 10       SUB 10            shift characters 30 to 38
D4F   FE 10       CP 10             look for '8' ie, '@' followed by 'space'
D51   CA 07* 0D*  JP Z D07
D54   FE 40       CP 40             look for '@'
D56   CA 07* 0D*  JP Z D07
D59   FE 0F       CP 0F             look for 'NL'
D5B   CA 6A 0D    JP Z D6A          jump SCROLL DOWN S.R.
D5E   FE 0D       CP 0D             look for 'back space'
D60   CA 00 0D    JP Z D00          clear screen
D63   00          NOP
D64   CD 3B 01    CALL CRT          print character
D67   C3 05 0D    JP D05            return to beginning of program
                  Scroll Down S.R:-
D6A   3E 20       LD A, 20
D6C   2A 18 0C    LD HL,(0C18 0C19)    remove '[illegible]' from screen
D6F   77          LD (HL), A
D70   00          NOP
D71   21 8A 0B    LD HL, 0B8A
D74   22 18 0C    LD 0C18, HL       place cursor bottom left
D77   21 79 0B    LD HL, 0B79
D7A   11 B9 0B    LD DE, 0BB9
D7D   01 80 03    LD BC, 380        scroll down
D80   ED B8       LDDR
D82   06 30       LD B, 30
D84   21 0A 08    LD HL, 080A
D87   77          LD (HL), A        eliminate top line of screen
D88   23          INC HL
D89   10 FC       DJNZ
D8B   C3 05 0D    JP D05            return to beginning
D8E   00 00       NOP
                  Cursor 'up/down' S.R:-
D90   2A 18 0C    LD HL, (0C18)
D93   01 40 00    LD BC, 0040
D96   72          LD (HL), D
D97   FE 2C       CP 2C             look for '<'
D99   28 04       JR Z +6
D9B   ED 4A       ADD HL, BC
D9D   ED 4A       ADD HL, BC        move cursor up or down one line
D9F   ED 42       SUB HL, BC
DA1   56          LD D, (HL)        store character 'covered' by cursor
DA2   1E 5F       LD E, 5F          load '[illegible]'
DA4   73          LD (HL), E        place '  ' on screen
DA5   22 18 0C    LD (0C18), HL     return new address of cursor
DA8   C3 07 0D    JP D07            return to beginning
```

Note : * can be changed to jump into alternative subroutines.

Christopher Oddy.

OPCODE DISPLAY

For Acorn users, this program will display machine codes in much the same way as the Intelligent Tabulator program does for the Nascom. On inspection of the Acorn manual it can be seen that there are areas of 1, 2 and 3 byte opcodes which can be easily separated by checking the two digits and subsequently testing for oddities like JSR.

Display Format

The program displays the least significant address byte (you only have 8 digits to play with) followed by the opcode and any operands. To make the display more legible dots are put on every other digit, thus splitting up the pairs of hex digits. The disassembler table can be broken down as follows :–

Single byte,	2nd digit	=	8 or A or,
	" "	=	0 and 1st digit < 8 and even.
Double byte,	2nd digit	=	1 to 6 or,
	" "	=	9 and 1st digit > 8 and odd.
Treble byte,	2nd digit	=	C to E or,
	" "	=	9 and 1st digit is odd.

Program Location

The program location is within the 128 bytes of RAM from 0E80 which is associated with the Port, but is completely relocatable.

```
0E80  A2 07     START  LDX =07        First Clear Display
0E82  94 10     CLEAR  STY X,10
0E84  CA               DEX
0E85  D0 FB            BNE CLEAR
0E87  A5 00            LDA Z,MAP      Get Opcode Address
0E89  20 6F FE         JSR DHEXTD     Display least significant
                                      byte on left
0E8C  A1 00            LDA (MAP,X)    Get the Opcode
0E8E  A0 02            LDY =02
0E90  20 6F FE         JSR DHEXTD     Display this on next pair
                                      of digits
0E93  A1 00            LDA (MAP,X)    Get it back and carry out
                                      disassembling
0E95  29 0F            AND =0F        Remove first 1st digit
```

```
0E97  F0 20          BEQ CHECK      if second digit = 0 check
                                    for 3 byters
0E99  C9 08          CMP #08
0E9B  90 0F          BCC 2B         if second digit < 8 we have
                                    a 2 byter
0E9D  F0 36          BEQ 1B         if second digit = 8 we have
                                    a 1 byter
0E9F  C9 0A          CMP #0A
0EA1  F0 32          BEQ 1B         if second digit = A we have
                                    a 1 byter
0EA3  B0 08          BCS 3B         if second digit > A we
                                    have a 3 byter
0EA5  A1 00          LDA (MAP,X)    Get it back again!
0EA7  29 10          AND #10        —we are left with 9's
0EA9  D0 02          BNE 3B         if first digit odd we have
                                    a 3 byter
0EAB  A2 02     2B   LDX #02        2 BYTE opcode
                                    (remember in X)
0EAD  88        3B   DEY            3 BYTE opcode
0EAE  B1 00          LDA (MAP),Y    get second byte
0EB0  A0 04          LDY #04
0EB2  20 61 FE       JSR DHEXTD     Display it
0EB5  8A             TXA            if X=0 we have a 3 byter
0EB6  D0 1F          BNE FINISH     —otherwise we have a 2
                                    byter
0EB8  A0 02          LDY #02
0EBA  B1 00          LDA (MAP),Y    get third byte
0EBC  A0 06          LDY #06
0EBE  20 61 FE       JSR DHEXTD     display it
0EC1  A2 03          LDX #03
0EC3  D0 12          BNE FINISH     3 byte opcode finished
0EC5  A1 00   CHECK  LDA (MAP,X)    Check for complicating
                                    opcodes--get opcode again
0EC7  C9 20          CMP #20        JSR?
0EC9  F0 E2          BEQ 3B
0ECB  29 F0          AND #F0        Remove second digit –
                                    leaving first!
0ECD  C9 80          CMP #80        greater than 8 --
0ECF  B0 DA          BCS 2B         2 Byter
0ED1  29 10          AND #10        check if odd or even
0ED3  D0 D6          BNE 2B         Odd–2 byter–otherwise–
0ED5  A2 01   1B     LDX #01        1 BYTE opcode–already
                                    finished!!
0ED7  E6 00   FINISH INC Z,MAP,X    increment MAP,X times –
0ED9  D0 02          BNE NO INC     – to move to next opcode
0EDB  E6 01          INC Z,MAPH,X
0EDD  CA      NOINC  DEX
0EDE  D0 F7          BNE FINISH     increment again
0EE0  A2 05          LDX #05        Now put dots on every
0EE2  A9 80   DOT    LDA #DOT       other digit to make it
0EE4  15 10          ORA Z,X,–10    more readable
0EE6  95 10          STA Z,X- 10
0EE8  CA             DEX
0EE9  CA             DEX
0EEA  10 F6          BPL DOT
0EEC  20 0C FE       JSR DISPLAY    Display disassembled
                                    opcode
0EEF  C9 16          CMP #UP        was UP (^) key pressed
0EF1  F0 8D          BEQ START      Yes -- carry on
0EF3  4C 09 FF       JMP SEARCH     No- Jump back to
                                    monitor
0EF6          END
```

1ST DIGIT / 2ND DIGIT

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BRK	ORD (I,X)				ORA ZERO	ASL ZERO		PHP	ORA IMMED	ASLA			ORA ABS	ASL ABS	
1	BPL	ORD (I),Y				ORD Z,X	ASL Z,X		CLC	ORA A,Y				ORA A,X	ASL A,X	
2	JSR	AND (I,X)			BIT ZERO	AND ZERO	ROL ZERO		PLP	AND IMMED	ROLA		BIT ABS	AND ABS	ROL ABS	
3	BMI	AND (I),Y				AND Z,X	ROL Z,X		SEC	AND A,Y				AND A,X	ROL A,X	
4	RTI	EOR (I,X)				FOR ZERO	LSR ZERO		PHA	EOR IMMED	LSRA		JMP ABS	EOR ABS	LSR ABS	
5	BVC	FOR (I),Y				EOR Z,X	LSR Z,X		CLI	EOR A,Y				EOR A,X	LSR A,X	
6	RTS	ADC (I,X)				ADC ZERO	FOR ZERO		PLA	ADC IMMED	RORA		JMP IND	ADC ABS	ROR ABS	
7	BVS	ADC (I),Y				ADC Z,X	ROR Z,X			ADC A,Y				ADC A,X	ROR A,X	
8		STA (I,X)			STY ZERO	STA ZERO	STX ZERO		DEY		TXA		STY ABS	STA ABS	STX ABS	
9	BCC	STA (I),Y			STY Z,X	STA Z,X	STX Z,Y		TYA	STA A,Y	TXS			STA A,X		
A	LDY IMMED	LDA (I,X)	LDX IMMED		LDY ZERO	LDA ZERO	LDX ZERO			LDA IMMED	TAX		LDY ABS	LDA ABS	LDX ABS	
B	BCS	LDA (I),Y			LDY Z,X	LDA Z,X	LDX Z,Y		CLV	LDA A,Y	TSX		LDY A,X	LDA A,X	LDX A,Y	
C	CPY IMMED	CMP (I,X)			CPY ZERO	CMP ZERO	DEC ZERO		INY	CMP IMMED	DEX		CPY ABS	CMP ABS	DEC ABS	
D	BNE	CMP (I),Y				CMP Z,X	DEC Z,X		CLD	CMP A,Y				CMP A,X	DEC A,X	
E	CPX IMMED	SBC (I,X)			CPX ZERO	SBC ZERO	INC ZERO		INX	SBC IMMED	NOP		CPX ABS	SBC ABS	INC ABS	
F	BEQ	SBC (I),Y				SBC Z,X	INC Z,X		SED	SBC A,Y				SBC A,X	INC A,X	

TELETEXT COLOUR

ANNOUNCING

VDU BOARD FOR

NASCOM 1 & 2

At last you can develop exciting colourful applications on your NASCOM

- Bring Computer Games to life.
- Display your Data in **Colour Graphics** form.
- Take your first step to a colour **Viewdata** Terminal

JUST SOME OF ITS FEATURES INCLUDE

- 13 Colours
- Full Viewdata/Teletext Character Set
- Alphanumerics and Graphics (5760 PELs)
- Flashing Characters
- Single or double height characters
- Plugs directly into NASBUS
- (No need to butcher your NASCOM!)
- On board PAL Modulator for direct connection to
- Colour TV aerial socket

Fully **ASSEMBLED AND TESTED**

£136 PLUS VAT

Money back Guarantee

ORDERS OR FULL DETAILS FROM

Winchester Technology Ltd.

PO Box 26. Eastleigh, Hants. SO5 5YY
Tel: 0415 66916

REACH INTO THE REAL WORLD WITH ANY NASCOM 1, NASCOM 2 AND MOST Z80 SYSTEMS.

"EFFORTLESS A TO D"

Low cost easily understood approach to A to D

APPLICATIONS:

Light Colour temperature
Bar code reader
Modem
Intelligent batch counter
Machine control
Measurement of velocity
Joysticks
Human stress analysis, etc, etc

Basic principles of the system are discussed and several applications dealt with in detail including hardware construction and software listings in assembler format

Manual **£4.75** + 30p p.&p.

Hardware kit including Transducers
£4.95 + 30p p.&p.

Still available Textie Mark Five word processor for Nascom 1 T4 **£13.50** + 25p p.&p.

THE SOFTWARE PUBLISHING Co.

8A Church Side, Mansfield, Notts.
Telephone: 0623 29237.
Access welcome

E PROMS

Surplus to requirements,

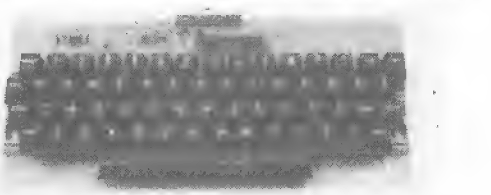
2708	**£6.00**
4716	**£25.00** each

Parts sent by return
Programming Service if required
Please add V.A.T. (15%) and 30p P & P

BARINGLOCK LTD

31 ASHDOWN AVENUE,
SALTDEAN,
BRIGHTON. BN2 8AH.
Tel. 01 686 1922

KB 060 ASC II KEYBOARD



60 keys in stepped rows. Auto repeat function. UC and LC ASCII coded. Brand new, built and tested. Rigid construction. With mating connector.

£44.75 (£53.19 inc P & P and VAT)

S.a.e. for data.

TIMEDATA Ltd. 57 Swallowdale, Basildon, Essex

"WE WON'T GET MUCH OUT OF THIS FOR THE NEXT FEW DAYS, — THIS IS A SICK NOTE!"

Trevor Lusty

Our regular problems continue with solutions to last months pot pourri

In the problems I set I try, if possible, to highlight the limitations as well as the capabilities of the computer. The problems last month showed how an apparently small increase in complexity can lead to a radical rethink of method.

Square And Add — 1

Well I did promise some short problems and you can't get one much shorter than this, can you? The reason for this problem's relative ease is the fact that only 9000 numbers need to be tested and all the results fit nicely within the range of a single real variable. Of course if you only have limited precision integer BASIC it's more difficult, but then, once you have solved this problem the next is only an extension. Figure 1 gives the program and solutions. I leave it to you to work out the details of this short program but I will comment that the solutions come in pairs — — — can you see how?

```
10 REM ******************************
20 REM *                            *
30 REM * PROGRAM --- SQUARE & ADD -1 *
40 REM *                            *
50 REM * PROGRAMMED IN 'PET' BASIC  *
60 REM *                            *
70 REM * TREVOR L LUSTY   30/12/1979 *
80 REM *                            *
90 REM ******************************
110 FOR N = 1000 TO 9999
120 LET S = N*N
130 LET F = INT( S/10000 )
140 LET L = INT( S - 10000*F + .5 )
150 IF  N = F+L  THEN PRINT N
160 NEXT N
170 END
READY.
```

2223	4941729
2728	7441984
4950	24502500
5050	25502500
7272	52881984
7777	60481729
9999	99980001

Fig.1. Square and Add the easy way.

Square And Add — 2

I had a problem myself when I set this one! Did I set a problem with which even those using extended BASIC would have difficulty, or did I settle for a more reasonable level of difficulty. Having already received a letter accusing me of being a sadist I decided to chicken out.

What makes this problem more difficult? Well firstly there is the question of representation, the square of a six digit number is just too big to fit into a single variable. (If you have extended BASIC just extend the problem.) We just have to find another way of storing the number, but fortunately a little simple algebra helps a lot. Now we all remember (said he sadistically) that

$$(a + b)(a + b) = axa + 2ab + bxb$$

and the important thing to realise as far as this problem is concerned is that if a is a number ending with 3 zeros then axa is a number ending with 6 zeros.

To solve the problem we don't actually want the complete square, all we need is the first six digits and the last six digits. The first six are the most significant digits in the square and may be found by the usual method, as shown in line 130 of figure 2. It is the second set of digits which will not be represented accurately and we use an algebraic trick to calculate these. The number under test is made up of two parts N1 and N2, where N = N1 + N2. N1 always ends with 3 zeros and therefore the square does not affect the values of the last six digits.

example :—

$$\begin{aligned} & 148149 \times 148149 \\ = {} & 148000 \times 148000 + 2 \times 148000 \times 149 + 149 \times 149 \\ = {} & 21904000000 + 44104000 + 22201 \\ = {} & 21948126201 \end{aligned}$$

```
10 REM ******************************
20 REM *                            *
30 REM * PROGRAM --- SQUARE AND ADD *
40 REM *                            *
50 REM * PROGRAMMED IN 'PET' BASIC  *
60 REM *                            *
70 REM * TREVOR L LUSTY    30.12.79 *
80 REM *                            *
90 REM ******************************
100 FOR N1 = 100000 TO 999000 STEP 1000
110 FOR N2 = 0 [illegible]
115 LET N = N1 + N2
120 LET S = N * N
130 LET F = INT( S/1000000 )
160 LET N3 = 2 * N1 * N2
170 LET L = INT( N3 - 1000000*INT( N3/1000000 )+.5 )
190 LET L = L + N2 * N2
200 LET L = INT( L - 1000000*INT( L/1000000 )+.5 )
210 IF  N = L+F  THEN PRINT N, F, L
220 NEXT N2
230 NEXT N1
240 END
READY.
```

142857	20408	122449
148149	21948	126201
181819	33058	148761
187110	35010	152100
208495	43470	165025
318682	101558	217124
329967	108878	221089
351352	123448	227904
356643	127194	229449
390313	152344	237969
461539	213018	248521
466830	217930	248900
499500	249500	250000
500500	250500	250000
533170	284270	248900
538461	289940	248521
609687	371718	237969
643357	413908	229449
648648	420744	227904
670033	448944	221089
681318	464194	217124
791505	626480	165025
812890	660790	152100
818181	669420	148761
851851	725650	126201
857143	734694	122449
999999	999998	1

Fig.2. Solution to the more difficult problem.

and we see that the digits 126201 come from the sum of the last two numbers only. Lines 160 – 200 of the program form these digits without reference to N1 squared. The flowchart (figure 3) helps to clarify the procedure.

The second factor which makes this problem different from the previous one is the time it takes. The 9000 numbers in the previous problem can be tested in 10 minutes or so. With 100 times that number to test this problem turns a coffee break into something approaching a weekend! If you used a less efficient algorithm you have my sympathy, if you saw the difficulty and used machine code then well done.

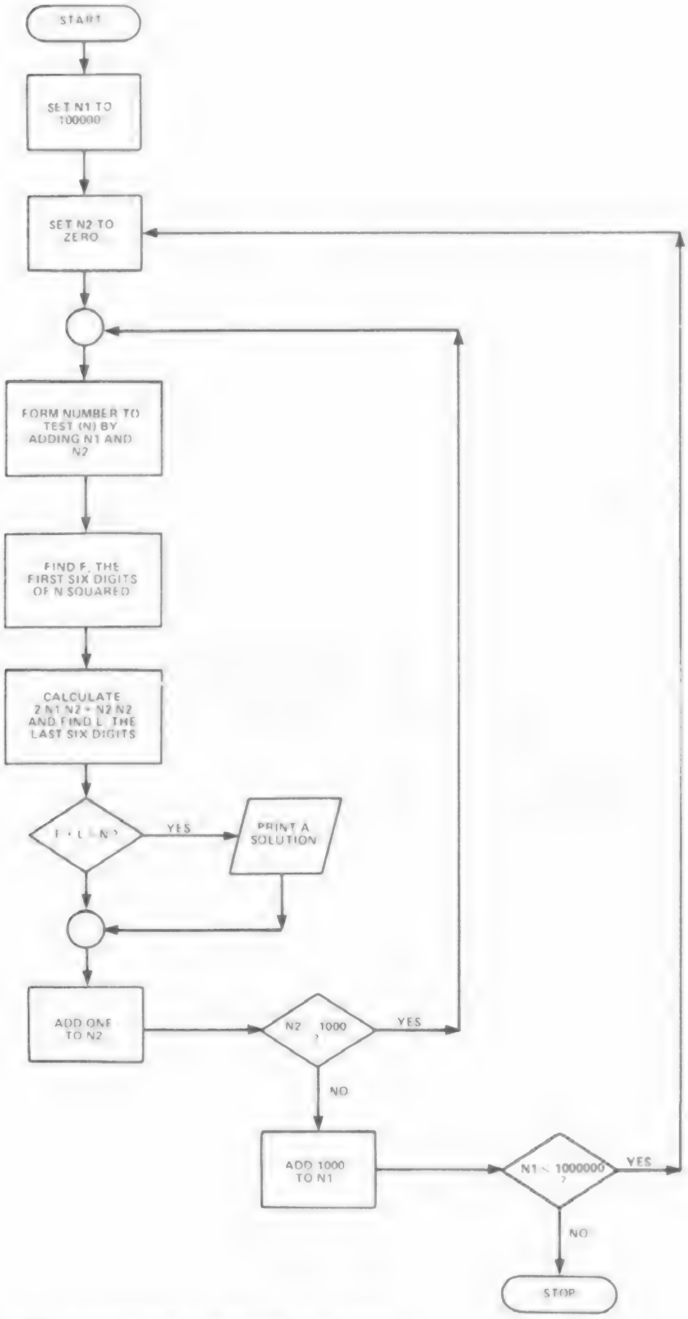


Fig.3. Flowchart for square and Add.

Powerful Digits

This was a simple example of nesting loops. The listing and run of the five digit problem is given in figure 4, I have not included the three digit listing as the program is so similar. However the answers are 153, 370, 371 and 407.

Problem Of The Month

Thank you for all your letters, I was amazed how far afield Computing Today is read and what a varied bunch you all are. Like most problem setters I enjoy a good problem myself and Fr. Curtis in Roscrea set me a beauty. He was using the Division Unlimited program to examine recurring decimals and could not find a fraction which recurred after 17 digits. He asked me to find a factor of 11,111,111,111, 111,111. Well I think that one's prime — — — of course, if you know different . . . . . .

Still, it made me think about finding the factors of fairly big numbers so :—

Hunt The Prime

Can you find the prime factors of the number :—

385,640,866,350,419 ?

and for all those whose letters started — — — 'with reference to Trevor Lusty's balls' I offer :—

Coconut Shy?

In a jungle clearing was a pile of coconuts in the form of a triangular pyramid. When the pyramid was dismantled it was found that the number of nuts formed an exact square. How many nuts were there in the pyramid?

```
10 REM ******************************
20 REM *                            *
30 REM * PROGRAM --- POWERFUL DIGITS *
40 REM *                            *
50 REM * PROGRAMMED IN  PET  BASIC  *
60 REM *                            *
70 REM * TREVOR L LUSTY    30/12/79 *
80 REM *                            *
90 REM ******************************
110 FOR A=1 TO 9
115 FOR B=0 TO 9
120 FOR C=0 TO 9
125 FOR D=0 TO 9
130 FOR E=0 TO 9
140 LET N= 10000*A + 1000*B + 100*C + 10*D + E
150 LET S = A*A*A*A*A + B*B*B*B*B + C*C*C*C*C
          + D*D*D*D*D + E*E*E*E*E
 160 IF  N <> S THEN 190
 170 PRINT N;"EQUALS THE SUM OF ITS DIGITS
 175 PRINT "RAISED TO THE FIFTH POWER
 180 PRINT
 190 NEXT E
 195 NEXT D
 200 NEXT C
 205 NEXT B
 210 NEXT A
 220 END
READY.


 54748 EQUALS THE SUM OF ITS DIGITS
RAISED TO THE FIFTH POWER

 92727 EQUALS THE SUM OF ITS DIGITS
RAISED TO THE FIFTH POWER

 93084 EQUALS THE SUM OF ITS DIGITS
RAISED TO THE FIFTH POWER
```

Fig.4. An example of nested loops.

J.G. Seal.

PICO BASIC

BASIC on an Mk14, you must be joking! Well here it is in all its glory to show you the way.

I bought an Mk14 kit at the beginning of 1979. After assembly it worked first time, and I went on to spend many happy hours familiarising myself with the programming code. I am a teacher, and it rapidly became clear to me that here was a machine that had great possibilities as an aid to the teaching of computing, in its early stages, but that there were three major problems which would have to be overcome.

a) The keyboard was very inefficient. Time and again addresses or data were misread.
b) The display was much too small, especially as I wished to be able to demonstrate what was happening to groups of pupils. However, I did not want to go to the expense of a VDU display system.
c) The available RAM was inadequate for something which it seemed to me would make the computer even more useful — the implementation of a simple high level number processing language.

As a first step towards overcoming these problems I replaced the keyboard with a set of small push switches (Maplin Ref. FF87U), the pins of which fitted exactly the holes in the PCB. However, I still got double bounce errors, and only overcame these finally when I removed from each switch the snap washer which gave it its 'click effect'.

Next, I replaced the miniature LED display strip with a display made up on a piece of Veroboard from eight 0.3 inch LED digits. This gave a readout which is easily viewed by several people at once.

Finally, I installed 1½K of additional RAM, addressed as indicated in the Mk14 instructions. The voltage regulator on the Mk14 proved able to supply this extra RAM, and also a tape interface module, but required a heatsink.

Language Development

I then turned my attention to the development of the high level language. None of the existing forms of BASIC would go into the memory I had available. This consisted of a total of 2K, but some of this was dedicated to the Mk14 monitor, and I wanted some for a data store. From what was left I had to find working space, program storage space, and space for the interpreter. It was clear that a compromise between all these demands was going to be difficult to work out.

I decided to attempt to devise a very simple subset of BASIC, and to call this micro-micro subset PICO BASIC. In this language the four basic arithmetic rules, + — x ÷, would be essential. The four BASIC instructions which seemed to me to be most necessary were INPUT, PRINT, GOTO, and an IF. And I wanted to be able to store and read data. It was clear that with a hexadecimal keyboard I would have to use single keys for each instruction, and would have to implement some form of upper and lower case system.

I decided from the beginning to aim at an integer arithmetic, and had hoped to cater for both negative and positive numbers. But as the system developed this had to go, and PICO now operates four-digit positive integer arithmetic, (mod 10,000). However, it is surprising how much useful computing can in fact be done within this limitation.

Descript Of The Language

The Mk14 keyboard is used as indicated in the table. Some of the keys are relabelled for convenience in use.

The PICO interpreter is entered from tape and stored as shown in the memory maps.

Overall Memory Map

0F00 — 0FFF		RAM
0E00 — 0EFF		(RAM I/O)
0D00 — 0DFF		DISPLAY
0C00 — 0CFF		(RAM I/O)
0B00 — 0BFF		RAM
0A00 — 0AFF		(RAM I/O)
0900 — 09FF		DISPLAY
0800 — 08FF		(RAM I/O)
0200 — 07FF		RAM
0000 — 01FF		MONITOR

Map of Available RAM

0FF7 — 0FFF	Monitor variables
0FAA — 0FF6	Serial data file
0F50 — 0FA9	Interpreter part C
0F12 — 0F4F	PICO variables
0F00 — 0F11	Monitor variables
0B00 — 0BFF	Interpreter part B
0600 — 07FF	PICO program store
0200 — 05FF	Interpreter part A

The 512 bytes of PICO program store allow the running of programs of up to 56 lines. The serial data file will hold 77 2-digit numbers, or, with a small change in the interpreter, 38 4-digit numbers.

General Form Of PICO Statements

n n x x x x x x x x

a) Line number nn in the range 01 to 99, two digits essential. Lines may be entered in any order, but will be executed in line number order.
b) Instruction x. . .x, up to 8 characters. If more than 8 characters are entered, the line aborts and must be re-entered, starting with the line number.

Entering PICO Programs

a) Address 021E.
b) Press RUN — display is ? followed by 7 blanks, indicating readiness to accept program line.
c) Enter program line. If an error is made, press RUN and start line again.
d) Press LINE — display indicates readiness for next line.
e) After entering last program line press END — display is RUN, indicating readiness to run. To run program press RUN.

Execution Of Programs

Statements are executed in the order of their line numbers, regardless of the order in which they were entered. If two or more lines have the same line number, only the last entered

is executed. A line can thus in effect be overwritten, and it may in effect be deleted by entering the line number on its own.

Error Detection And Correction

Error detection is written into the interpreter. When the computer attempts to execute at run time a line containing an error in syntax, the run halts and ? is displayed. The number of the line which caused the halt can be discovered by the following procedure.

a) Return to monitor – press X.
b) Address 0E1A – the two most right digits show the required line number.

To see what is in that line, carry out the following.

c) Address 0F1D and enter the line number at that address.
d) Address 0339 and press RUN – display is RUN plus a set of symbols (related to the segment code for the line number).
e) Press RUN. The required line is displayed.

To alter the line, proceed as follows. (If no alteration is required, jump to step j below.

f) Press RUN – display is ?
g) Enter revised line, including line number.
h) Press LINE.
i) Press END – display is RUN plus the symbols, which serve to remind of the necessity to reset 0F1D to zero.
j) Press X.
k) Address 0F1D and enter zero (or return to step c above to see another line).
l) Address 0339.
m) Press RUN.

If it is desired to alter a line without the necessity of seeing it first, follow these steps.

n) Address 0339 and press RUN.
o) Press INPUT.
p) Proceed as from c) in the section 'Entering PICO programs'.

PICO Statements

INPUT eg 01Inv4

To enter this press 01 Λ INPUT Λ 4

At run time this statement produces the display v4=?, and the run halts to await entry of a number. If more than 4 digits are entered, only the last 4 are retained.

After the number has been entered, press RUN to resume execution of the program. The number will be stored in variable v4, one of ten available variables which are designated v0 to v9.

PRINT eg 02Prv5

At run time this statement produces the display v5=xxxx, the number stored in v5. Leading zeroes are suppressed.

Press RUN to continue execution.

GOTO eg G006

This causes the program to jump immediately to line 06. If the destination line number does not exist in the program, execution will jump to the first existing one above it, or, failing that, to the start of the program.

IF eg (04)IFv5v608

As the last two characters are entered, the line number disappears off the left of the display.

This statement is equivalent to the normal BASIC statement.

04 IF v5 ⩾ v6 THEN 08

It causes execution to move immediately to line 08 *if* the number in v5 is greater than or equal to the number in v6. If v5 is less than v6 execution continues at the next higher line number above 04.

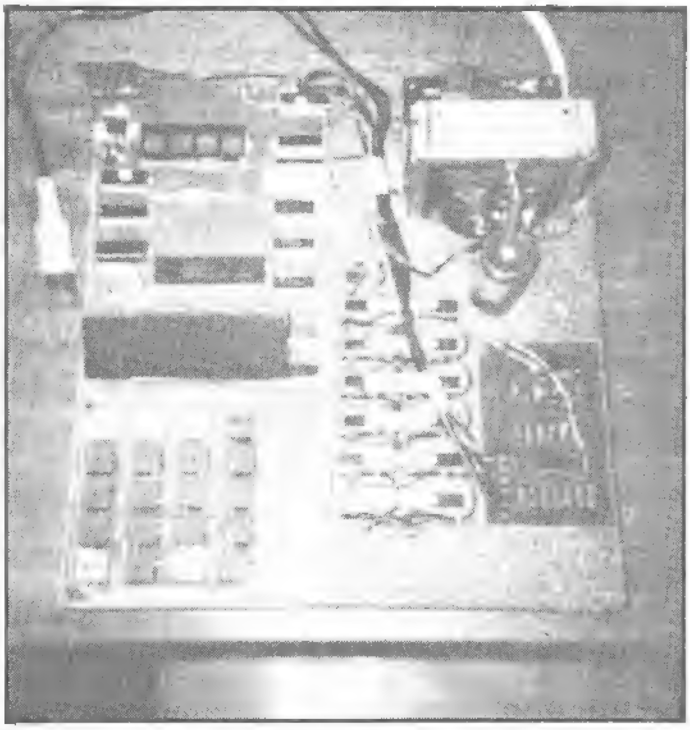
By reversing the positions of v5 and v6 the effect v5 ⩽ v6 is obtained. This is the most useful single conditional jump statement.

ASSIGN CONSTANT eg 05v7=xxxx

This causes the number xxxx to be stored in v7.

ASSIGN VARIABLE eg 06v8=v7

This causes the contents of v7 to be stored also at v8.



The author's Mk14. The power supply unit is at top right, the tape interface module at lower right, and the additional RAM between that and the main PCB. On the PCB can be seen the heat sink (top left), the new display board, and the re-labelled keyboard. The tape interface LEDs have been included on the display board.

Calculate

The four functions add, subtract, multiply and divide may be invoked, but unlike conventional BASICs only one calculation may take place per program line.

In the example below f stands for any one of the four functions.

(07)v1=afb

This is the general form, and a,b stand for numbers or variables. A maximum of 5 display digits is available for afb. When both a and b are variables these are fully used.

eg (07)v1=v1+v2

When a or b is a variable and the other is a number, the number may not have more than 2 digits.

eg (07)v1=v2–23

When both a and b are numbers, the total number of their digits may not exceed 4. When it is desired to use larger numbers they must first be assigned to variables.

Notes on the calculations

a) Answers are given mod 10,000. Thus 11,340 would be shown as 1,340, and –2 as 9,998.
b) Multiplication is effected by repeated addition without shifts, and 27x932 takes longer than 932x27.
c) The answer to a division gives the integer part of the quotient – the answer is truncated, not rounded.

Input & Load Data

A serial data file for up to 77 2-digit numbers is available. A modification of the interpreter (given in the listing) will change this to 38 4-digit numbers if required.

During each program run the numbers are normally accessed in order from the beginning of the file using these statements.

```
eg 08Idv9
   09Ldv6
```

The first statement causes the two right hand digits of the number in v9 (the whole number if the 4-digit file is in use) to be stored at the next file location. The second causes the contents of the next file location to be stored at v6.

The number of the file location to be next accessed is held in v0 in hexadecimal. It is incremented by 1 after each access. By storing a decimal number from 00 to 49 in v0, 50 of the file locations can be directly accessed at any point of a program. By treating the two figures of this number as array subscripts it is possible to implement a 5x10 array.

Data may be entered during one program run and retrieved at a later date by a different program, providing that the computer has not been switched off. It can also, of course, be transferred to tape for permanent storage.

Notes On Programming

a) When the program linefile is full (contains 56 lines) any attempt to enter a further line produces the display F.

b) The 56 includes any lines which have been repeated to correct errors or to vary the program.

c) A routine to display any number of decimal places resulting from a division is included among the sample programs.

d) There are 10 variables denoted v0 to v9. v0 doubles as data file address store.

e) At the start of every run these variables are zeroed.

Sample Programs

MULTIPLES

This simple program displays successive multiples of a chosen number.

```
01 In vl
02 v0=v0+v1
03 Pr v0
04 GO 02
```

Note the closed loop with the PRINT statement in it. In conventional BASIC this would produce an endless succession of multiples which could only be halted by breaking into the program run. In PICO the run halts each time the PRINT statement is reached, to be resumed by pressing RUN. But as in normal BASIC the only way to get out of the loop is by returning to the monitor.

DATA INPUT

This loop is used to enter data into the serial data file.

```
05 In vl
10 Id vl
15 GO 05
```

MEAN OF NUMBERS IN DATA FILE

The program includes the use of a stop value (rogue value) at the end of the data to get out of the load loop.

```
01 v1=99          Stop value = 99
02 Ld v2
03 IF v2 v1 07
04 v3=v3+1
05 v4=v4+v2
06 GO 02
07 v5+v4/v3
08 Pr v4          Displays total of numbers
09 Pr v3          Displays number of numbers
10 Pr v5          Displays mean of numbers
```

KEY	LOWER CASE		UPPER CASE	
	LABEL	DISPLAY	LABEL	DISPLAY
0–9	0–9	0–9	0–9	v0–v9
A	+	-/	GOTO	GO
B	–	–	IF	IF
C	*	.		Ld
D	/	/-/	PRINT	Pr
E	=	=	INPUT	In
F	^	v		Id
TERM	END	End of program, prepare run		
MEM	LINE	End of line, store in program		
GO	RUN			
ABORT	X	Return to monitor		

Notes :
a C upper case is for LOAD (READ) DATA
b F lower case is for shift to upper case
c F upper case is for INPUT (WRITE) DATA

Key designations for PICO BASIC on the Mk 14 keyboard.

DIVISION WITH DECIMAL PLACES

The first run produces the integer part of the answer. Each successive press of RUN produces one decimal place of the answer.

```
01 In v1
02 In v2
03 v3=v1/v2
04 Pr v3
05 v4=v3xv2
06 v5=v1–v4
07 v1=v5x10
08 GO 03
```

SQUARE ROOT

The integer part of the square root of a given number is produced by the usual iterative method.

```
01 v0=10
02 v1=10
03 v4=0
04 In v2          Enter number whose root is required
05 v3=v2/v1
06 v3=v3+v1
07 v1=v3/2
08 v4=v4+1
09 IF v0 v4 05
10 Pr v1
11 GO 02
```

Interpreter Listing

There follows a listing of the interpreter. It is highly probable that by careful consideration of each section of the interpreter it will be possible to prune it and thus release extra memory for the inclusion of desirable refinements such as a simplified way of displaying any line of the program in store.

```
0200            Conversion codes
                Display
  00  3F        0
  01  06        1
  02  5B        2
  03  4F        3
  04  66        4
  05  6D        5
  06  7D        6
  07  07        7
  08  7F        8
  09  67        9
  0A  46        /  +
  0B  40
  0C  80           x
  0D  52        ~  ÷
  0E  48        =
  0F  3D        G
  10  06        I
  11  38        L
  12  73        P
  13  06        I
  14  06        I
  15  3F        0
  16  71        F
  17  5E        d
  18  50        r
  19  54        n
  1A  5E        d
  1B  00        BLANK
  1C  1C        v
  1D  53        ?
```

```
021E            Set P2, clear all stores 0F13 - 4F, set
                PAK H to 06 New program entry
  1E  C4 0F
  20  36
  21  C4 00
  23  32        P2 set 0F00
  24  C4 2D
  26  CA 12     K=2D
  28  02
  29  F4 12     LOOP 1 ADI 12; OFFSET
  2B  01
  2C  C4 00
  2E  CA 80     0F4F - 13 = 0
  30  BA 12     DLD K
  32  9C F5     JNZ LOOP 1
  34  C4 06
  36  CA 1C     PAK H = 06
```

```
0238            Set LSK to A, set LINE to BLANK,
                clear UCF, ELF, set L7 to ?
                New line entry
  38  C4 0A
  3A  CA 16     LSK=A
  3C  CA 12     K=A
  3E  02
  3F  F4 1F     LOOPS 2 ADI 1F - OFFSET
  41  01
  42  C4 1B
  44  CA 80     L9 - L0=BLANK
  46  BA 12     DLD K
  48  9C F8     JNZ LOOP 2
  4A  C4 00
  4C  CA 14     UCF=0
  4E  CA 15     ELF=0
  50  C4 1D
  52  CA 27     L7=?
```

```
0254            Convert LINE to SEG and DISPLAY
                New character entry
  54  C4 02
  56  35
  57  C4 00
  59  31        P1 set 0200
  5A  C4 0F
  5C  37
  5D  C4 00
  5F  33        P3 set 0F00
  60  C4 08
  62  CB 12     K=8
  64  C2 27     LOOP 3 LD L7 - L0
  66  01
  67  C1 80     LD SEG code
  69  CA 07     ST SEG7 - 0
  6B  C6 FF     @-1 P2
  6D  BB 12     DLD K
  6F  9C F3     JNZ LOOP 3
  71  C6 08     @+8 P2
  73  02
  74  C4 01
  76  37
  77  C4 84
  79  33
  7A  3F        DISPLAY
```

```
027B            Return, divert COMM, shift LINE left,
                process N,L
  7B  90 60     JMP 02DD with COMM
  7D  C4 0F     (N,L in E)
  7F  37
  80  C4 00
  82  33        P3 set 0F00
  83  C4 09
  85  CB 12     K=9
  87  C2 28     LOOP 4 LD L8 - L0
  89  CA 29     ST L9 - L1
  8B  C6 FF     @-1 P2
  8D  BB 12     DLD K
  8F  9C F6     JNZ LOOP 4
  91  C6 09     @+9 P2
  93  BA 16     DLD LSK   neg if more than 8 shifts
  95  94 02     JP 0299
  97  90 9F     JMP 0238 New line entry, line abort
  99  40
  9A  E4 0F     E=F?
  9C  9C 0C     JNZ 02AA
  9E  C4 1C
  A0  CA 20     L0=v
  A2  AA 14     ILD UCF
  A4  E4 02     UCF=2?
  A6  98 29     JZ 02D1
  A8  90 AA     JMP 0254 New character entry
  AA  C2 14     ILD UCF
  AC  E4 01     UCF=1?
  AE  98 10     JZ UCS at 02C0
  B0  40        LD E (N,L)
  B1  CA 20     L0=N,L
  B3  C2 15     LD ELF (1 if line END)
  B5  98 9D     JZ 0254 New character entry C
  B7  C2 16     LD LSK (0 if line left - justified)
  B9  98 32     JZ 02ED Store line in program
  BB  C4 1B
  BD  01        E=BLANK
  BE  90 BD     JMP 027D to complete left justify
```

```
02C0            UCS - store upper case symbols
  C0  CA 14     UCF=0
  C2  40        LD E (N,L)
  C3  02
  C4  F4 F6     ADI F6 (- A)
  C6  94 09     JP 02D1 - if E is a letter
  C8  C4 1C
  CA  CA 21     L1 - v
  CC  40
  CD  CA 20     L0=N
  CF  90 83     JMP C
  D1  40
  D2  02
  D3  F4 05
  D5  CA 21
  D7  F4 06
  D9  CA 20
  DB  90 F2     JMP C
```

```
02DD            Process COMM
  DD  40        LD E (COMM)
  DE  E4 22     E=22 (RUN)?
  E0  98 B5     JZ 0297 - Line abort
  E2  40
  E3  E4 27     E=27 (END)?
  E5  98 52     JZ 0339 - RUNSET
  E7  AA 15     ILD ELF
  E9  90 C8     JMP 02B3 to left justify before store
```

```
02EB            Jump point
  EB  90 AA     JMP 0297 for new line and line abort
```

```
02ED            STORE LINE in PICO program, store
                maximum LINO in MLN
  ED  C2 29     LD L9
  EF  E4 1B     L9=BLANK
  F1  98 A4     JZ 0297 for new line
  F3  C4 0F
  F5  35
  F6  C4 20
  F8  31        P1 set L0
  F9  C2 1C
  FB  37
  FC  C2 1B
  FE  33        P3 set PAK - start of new PICO line
  FF  C2 1B     LD PAK L
0301  02
  02  F4 09     ADI 9
  04  CA 1B     ST PAK L
  06  C2 1C     LD PAK H
  08  F4 00     ADI 0
  0A  CA 1C     ST PAK H
  0C  E4 08     PAK H=8?
  0E  98 23     JZ 0333 - PICO line store full
  10  C4 08
  12  CA 12     K=8
0314  C5 01     LOOP 5 LD L0 - L7, @+1 P1
  16  CF 01     ST @+1 P3
  18  BA 12     DLD K
  1A  9C F8     JNZ LOOP 5
  1C  C2 29     LD L9
  1E  1E 1E
  20  1E 1E     SL4
  22  02
  23  F2 28     ADD L8 - LINO in ACC
  25  CF 01     ST @+1 P3
  27  02
  28  FA 13     CAD MLN
  2A  06
  2B  94 04     JP 0331
  2D  C3 FF     LD LINO
  2F  CA 13     ST MLN
  31  90 B8     JMP 02EB for new line entry
  33  C4 16
  35  CA 27     L7=F
  37  90 96     JMP C
```

```
0339            RUNSET
  39  C4 0F
  3B  36
  3C  C4 00
  3E  32        P2 set 0F00
  3F  C4 50
  41  CA 07     ST SEG r L7
  43  C4 1C
  45  CA 06             u L6
  47  C4 54
  49  CA 05             n L5
  4B  C2 1D     LD DLK
  4D  CA 04     L4 BLANK or symbol for line display
  4F  CA 03     L3  "  "  "  "  "  "
  51  CA 02     L2  "  "  "  "  "
  53  CA 01     L1  "  "  "  "  "  "
  55  CA 00     L0  "  "  "  "  "  "
  57  CA 1A     PLI - 0
  59  C4 20
  5B  CA 12     K='20
  5D  02
  5E  F4 2F     LOOP 6 ADI  OFFSET
  60  01
  61  C4 00
  63  CA 80     0F4F - 30=0
  65  BA 12     DLD K
  67  9C F5     JNZ LOOP 6
  69  02
  6A  C4 01
  6C  37
  6D  C4 84
  6F  33
  70  3F        DISPLAY
```

```
0371  90 02     JMP 0375
  73  90 BC     JMP 0331 for new line entry
```

```
0375            SCAN PICO file for required line
  75  C2 1C
  77  37
  78  C2 1B
  7A  33        P3 set PAK
  7B  C3 FF     LOOP 7 LD LINO
  7D  E2 1A     LINO=PLI?
  7F  98 1E     JZ 03A5 - EXECUTE
  81  33        P3 in ACC
  82  01        P3 in E
  83  40        P3 in ACC
  84  E4 09     P3=9?
  86  98 06     JZ 038E to re SCAN
  88  40
  89  33
  8A  C7 F7     @-9 P3
  8C  90 ED     JMP LOOP 7
  8E  C2 1A     LD PLI
  90  E2 13     PLI=MLN?
  92  98 A5     JZ 0339 RUNSET
  94  C2 1A     LD PLI
  96  02
  97  EC 01     DAI 1
  99  CA 1A     ST PLI
  9B  90 D8     JMP 0375 SCAN (a)
```

```
039D            Jump point
  9D  90 92     JMP 0331 for new line entry
```

```
039F            EXECUTE - Transfer PICO line to
                LINE. Identify statement.
  9F  C4 0F
  A1  35
  A2  C4 29
  A4  31        P1 set L9 (P3 set LINO+1)
  A5  C4 09
  A7  CA 12     K=9
  A9  C7 FF     LOOP 8 @-1 P3 LD PICO line
  AB  CD FF     @-1 P1 ST L8 - L0
  AD  BA 12     DLD K
```

```
AF    9C F8    JNZ LOOP 8
B1    C2 1D    LD DEK
B3    9C 82    JNZ 0337 for C
B5    00 00    Spare
B7    C2 27    LD L7
B9    02
BA    F4 E4    ADI E4 (-1C)
BC    94 6A    JP 0428 for ASSIGN
BE    F4 01    JP 03BE for SCAN (b) (Blank line)
C0    94 CC
C2    F4 07
C4    94 6A    JP 0430 for INPUT (WRITE) DATA
C6    F4 01
C8    94 60    JP 042A for INPUT
CA    F4 01
CC    94 5E    JP 042C for PRINT
CE    F4 01
D0    94 5C    JP 042E for LOAD (READ) DATA
D2    F4 01
D4    94 15    JP 03EB for IF
D6    F4 01
D8    94 02    JP 03DC for GOTO
DA    90 C1    JMP 039D for new line entry
```

```
03DC           GOTO
DC    C2 25    LD L5
DE    1E 1E
E0    1E 1E    SL4
E2    02
E3    F2 24
E5    CA 1A    ST PLI
E7    90 8C    JMP 0375 for SCAN (a)
```

```
03E9           Jump point
E9    90 A3    JMP 038E SCAN (b)
```

```
03EB           IF
EB    C4 0F
ED    37
EE    C4 00
F0    33       P3 set 0F00
F1    C4 02
F3    CA 12    K=2
F5    C2 22    LOOP 9 LD L2
F7    02
F8    F4 40    ADI 40
FA    01
FB    C3 80    LD var 2 H,L
FD    CA 17    ST TS1
FF    C2 2[illegible]    LD L[illegible]
0401  02
02    F4 40
04    01
05    C3 80
07    03
08    FA 17    CAD var 2 H,L
[illegible]    JNZ 0414
[illegible]       10 P3
[illegible] BA [illegible]    DLD K
10    9C E3    JNZ LOOP 9
12    90 03    JMP 0417
14    06
15    94 0D    JP 0424
17    C2 21    LD L1
19    1E 1E
1B    1E 1E    SL4
1D    02
1E    F2 20    ADD L0
20    CA 1A    ST PLI
22    90 C3    JMP 03E7 for SCAN (a)
24    90 C3    JMP 03E9 for SCAN (b)
```

```
0426           Jump points
26    90 B2    JMP 03DA for new line entry
28    90 60    JMP 048A for ASSIGN
2A    90 62    JMP 048E INPUT
2C    90 5E    JMP 048C for PRINT
2E    90 2C    JMP 045C LOAD (READ) DATA
```

```
0430           INPUT (WRITE) DATA
30    C2 25    LD L5
32    E4 1C    L5=v?
34    9C A4    JNZ 03DA for new line entry
36    C4 0F
38    35
39    C4 A9
3B    31       P1 set 0F A9 (Start of data file at
               0F AA)
3C    C2 24    LD L4
3E    02
3F    F4 30    ADI 30
41    01
42    C2 80    LD var L
44    01       var L in E
45    AA 30    ILD DAK
47    01       DAK in E, var L in ACC
48    C9 80    ST var L at DAK
4A    90 0E    JMP 045A for SCAN (b)
```

```
[illegible]    LD var H
[illegible]    var H in E
[illegible] AA [illegible]    ILD DAK
[illegible]    DAK in E, var H in ACC
[illegible]    ST var H at DAK
[illegible]    ILD DAK
[illegible]    DAK in E
[illegible] C2 17    LD TS1 - var L
[illegible] C9 80    ST var L at DAK
[illegible] 90 8D    JMP 03E9 for SCAN (b)
```

044C to 0459 not used, as data is set for 2-figure numbers. To change to 4 figure numbers alter 0442 to 044B as shown in appendix

```
045C           LOAD (READ) DATA
5C    C2 25    LD L5
5E    E4 1C    L5=v?
60    9C C4    JNZ 0426 for new line entry
62    C4 0F
64    [illegible]
65    C4 A9
67    31       P1 set 0F A9 (Start of data file at
               0F AA)
68    AA 30    ILD DAK
6A    01
6B    C1 80
6D    01
6E    C2 24    LD L4
70    02
71    F4 30    ADI 30
73    01
[illegible]    ST data at var L
[illegible]    JMP 0486 for SCAN (b)
[illegible]    ADI 30
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]    JMP 0424 for SCAN (b)
```

0478 to 0485 not used, as data is set for 2-figure numbers. To change to 4-figure numbers alter 046B to 0477 as shown in appendix

```
0488           Jump points
88    90 9C    JMP 0426 for new line entry
8A    90 6E    JMP 04FA for ASSIGN
8C    90 6E    JMP 04FC for PRINT
```

```
048E           INPUT
8E    C2 25    LD L5
90    E4 1C    L5
92    9C 92    JNZ 0426 for new line entry
94    C4 [illegible]
96    CA 27    L7
98    C2 24    LD L4
9A    CA 26    L6 L4
9C    C4 0F
9E    CA 25    L5
A0    C4 1D
A2    CA 24    L4 ?
A4    C4 1B
A6    CA 23    L3 BLANK
A8    CA 22    L2- "
AA    CA 21    L1
AC    CA 20    L0
AE    C4 0F
B0    37
B1    C4 00
B3    33       P3 set 0F00
B4    C4 02
B6    35
B7    C4 00
B9    31       P1 set 0200
BA    C4 08
BC    CB 12    K=8
BE    C2 27    LOOP 10 LD L7 - L0
C0    01
C1    C1 80    LD SEG CODE
C3    CA 07    ST "    "
C5    C6 FF    @-1 P2
C7    BB 12    DLD K
C9    9C F3    JNZ LOOP 10
CB    C6 08    @+8 P2
CD    02
CE    C4 01
D0    37
D1    C4 84
D3    33
D4    3F       DISPLAY
```

```
04D5  90 27    JMP 04FE COMM
D7    C4 0F
D9    37
DA    C4 00
DC    33       P3 set 0F00
DD    C4 1B
DF    CA 24    L4=BLANK
04E1  C4 03
E3    CB 12    K=3
E5    C2 22    LOOP 11 LD L2 - L0
E7    CA 23    ST L3 - L1
E9    C6 FF    @-1 P2
EB    BB 12    DLD K
ED    9C F6    JNZ LOOP 11
EF    C6 03    @+3 P2
F1    40       LD E (N,L)
F2    CA 20    ST L0
F4    90 BE    JMP 04B4 for C
```

```
04F6           Jump points
F6    90 8E    JMP 0486 for SCAN (b)
F8    90 8E    JMP 0488 for new line entry
FA    90 45    JMP 0541 for ASSIGN
FC    90 45    JMP 0543 PRINT
```

```
04FE  C4 0F    COMM
0500  37
01    C4 00
03    33       P3 set 0F00
04    C4 0F
06    35
07    C4 00
09    31       P1 set 0F00
0A    C2 26    LD L6
0C    02
0D    F4 40    ADI 40
0F    01
10    C4 04
12    CB 12    K=4
14    C2 23    LOOP 12 LD L3 - L0
16    E4 1B    L3 - L0=BLANK
18    9C 02    JNZ 051C
1A    CA 23    L3 - L0=0
1C    C6 FF    @-1 P2
1E    BB 12    DLD K
20    9C F2    JNZ LOOP 12
22    C6 04    @+4 P2
24    C4 02
26    CB 12    K=2
28    C2 23    LOOP 13 LD L3,L1
2A    1E 1E
2C    1E 1E    SL4
2E    02
2F    F2 22    ADD L2,L0
31    C9 80    ST var H,L
33    C6 FE    @-2 P2
35    C5 F0    @-10 P1
37    BB 12    DLD K
39    9C ED    JNZ LOOP 13
3B    C6 04    @+4 P2 set 0F00
3D    90 B7    JMP 04F6 for SCAN (b)
```

```
053F           Jump points
3F    90 B7    JMP 04F8 for new line entry
41    90 7A    JMP 05BD for ASSIGN
```

```
0543           PRINT
43    C2 25    LD L5
45    E4 1C    L5=v?
47    9C AF    JNZ 04F8 for new line entry
49    C4 1C
4B    CA 27    L7=v
4D    C2 24    LD L4
4F    CA 26    L6 L5
51    C4 0E
53    CA 25    L5= =
55    C4 1B
57    CA 24    L4=BLANK
59    C4 0F
5B    37
5C    C4 [illegible]
5E    33       P3 set 0F00
5F    C4 0F
61    35
62    C4 00
64    31       P1 set 0F00
65    C2 26    LD L6
67    02
68    F4 40    ADI 40
6A    01
6B    C4 02
6D    CB 12    K=2
6F    C1 80    LOOP 14 LD var H,L
71    D4 0F    Select right digit
73    CA 22    ST L2,L0
75    C1 80    LD var H,L
77    1C 1C
79    1C 1C    SR4
7B    CA 23    ST L3,L1
7D    C6 FE    @-2 P2
7F    C5 F0    @-10 P1
81    BB 12    DLD K
83    9C EA    JNZ LOOP 14
```

```
  85   C6 04   @+4 P2
  87   C4 02
  89   35
  8A   C4 00
  8C   CA 19   LZF=0
  8E   31      P1 set 0200
  8F   C4 08
  91   CB 12   K=8
  93   C3 12   LOOP 15 LD K
  95   E4 01   K=1?
  97   98 11   JZ 05AA
  99   C3 12
  9B   02
  9C   F4 FB   ADI FB (--5)
  9E   94 0A   JP 05AA
  A0   C3 19   LD LZF
  A2   F2 27   ADD L3
  A4   CB 19   ST LZF
  A6   9C 02   JNZ 05AA
  A8   90 05   JMP 05AF
  AA   C2 27   LD L7
  AC   01
  AD   C1 80
  AF   CA 07
  B1   C6 FF   @--1 P2
  B3   BB 12   DLD K
  B5   9C DC   JNZ LOOP 15
  B7   90 06   JMP 05BF
```

```
05B9           Jump points
  B9   90 82   JMP 053D for SCAN (b)
  BB   90 82   JMP 053F for new line entry
  BD   90 0E   JMP 05CD ASSIGN
```

```
05BF   C6 08   @+8
  C1   02
  C2   C4 01
  C4   37
  C5   C4 84
  C7   33
  C8   3F      DISPLAY
```

```
  C9   90 00
  CB   90 EC   JMP 05B9 for SCAN (b)
```

```
05CD           ASSIGN
  CD   C2 25   LD L5
  CF   E4 0E   L5= =?
  D1   9C E8   JNZ 05BB for new line entry
  D3   C4 0F
  D5   35
  D6   C4 20
  D8   31      P1 set 0F20 -- LINE
  D9   C2 24   LD L4
  DB   E4 1C   L4=v? var (or const)?
  DD   9C 07   JNZ 05E6 if const
  DF   C4 0B
  E1   37
  E2   C4 47
  E4   33
  E5   3F      JMP 0B48 if var
  E6   C4 04
  E8   CA 12   K=4
  EA   01      LOOP 16 K in E
  EB   C1 80   LD L(K)
  ED   02
  EE   F4 F6   ADI F6 (-A) L(K)=m?
  F0   94 06   JP 0B00 if 1(K) not n
  F2   BA 12   DLD K
  F4   94 F4   JP LOOP 16
  F6   90 C3   JMP 05BB for new line entry if 5 figs
  F8   C4 0A
  FA   37
  FB   C4 FF
  FD   33 3F   JMP 0B00
  FF   00      Spare

0B00   C2 12   LD K
  02   CA 17   ST TS1, K for FUNC
  04   E4 04   K=4 still?
  06   9C 07   JNZ 0B0F to continue
  08   C4 02
  0A   37
  0B   C4 96
  0D   33
  0E   3F      JMP 0297 for new line entry
  0F   C4 0F
  11   37
  12   C4 00
  14   CA 4A
  16   33      P3 set 0F00
  17   C4 02
  19   CA 18   K1(TS2)=2
  1B   AA 12   LOOP 17 ILD K
  1D   E4 04   K=4?
  1F   9C 09   JNZ 0B2A
  21   C2 12   LD K
  23   01      K in E
  24   C1 80   LD L(K)
  26   CB 3A   ST var AL,AH (1,3 figs)
  28   90 34   JMP 0B5E
  2A   AA 12   ILD K
  2C   01
  2D   C1 80   LD L(K)
  2F   1E 1E
  31   1E 1E   SL4
  33   01      n in E
  34   BA 12   DLD K
  36   01      K in E, n in ACC
  37   02
  38   F1 80   ADD L(K)
  3A   CB 3A   ST var AL,AH (2,4 figs)
  3C   AA 12   ILD K
  3E   E4 04   K=4?
  40   98 1C   JZ 0B5E
  42   C7 10   @+10 P3
  44   BA 18   DLD K1
  46   9C D3   JNZ LOOP 17
  48   C2 23   LD L3
  4A   02
  4B   F4 30
  4D   01
  4E   C2 80   LD var L
  50   CA 3A   ST var AL
  52   01
  53   F4 10
  55   01
  56   C2 80   LD var H
  58   CA 4A   ST var AH
  5A   C4 02
  5C   CA 17   ST TS1 -- K for function
  5E   C2 1E   LD ASF
  60   E4 01   ASF=1?
  62   98 50   JZ 0BB4, SELECT
  64   C4 01
  66   CA 1E   ASF=1
  68   C2 17   LD K(FUNC)
  6A   01      K(FUNC) in E
  6B   C1 80   LD FUNC
  6D   CA 1F   ST FS
  6F   E4 1B   FUNC=Blank?
  71   9C 1D   JNZ 0B90
  73   C4 00
0B75   CA 1E   ASF=0
  77   C2 26   LD L6
  79   02
  7A   F4 30
  7C   01
  7D   C2 3A   LD var L
  7F   CA 80   ST var L
  81   01
  82   F4 10
  84   01
  85   C2 4A   LD var H
  87   CA 80   ST var H
  89   C4 03
  8B   37
  8C   C4 8D
  8E   33
  8F   3F      JMP 038D SCAN (b)
  90   C2 3A
  92   CA 3B
  94   C2 4A
  96   CA 4B
  98   C2 17   LD K(FUNC)
  9A   CA 12   K=K(FUNC)
  9C   BA 12   DLD K
  9E   01      K in E
  9F   C1 80   LOOP 18 LD L(K)
  A1   C9 04   ST L4
  A3   C4 1B
  A5   C9 03   ST Blank L3
  A7   C5 FF   @-1 P1
  A9   BA 12   DLD K
  AB   94 F2   JP LOOP 18
  AD   C4 05
  AF   37
  B0   C4 D2
  B2   33
  B3   3F      JMP 05D3 to repeat (1),(2)
```

```
0BB4           SELECT
  B4   C2 1F   LD FS
  B6   01      FS in E
  B7   40
  B8   E4 0D   FUNC is divide?
  BA   98 3C   JZ 0BF8 for divide
  BC   40
  BD   E4 0C   FUNC is multiply?
  BF   9C 30   JNZ 0BF1 for +
  C1   C2 3A
  C3   CA 3C
  C5   C2 4A
  C7   CA 4C
  C9   C4 00
  CB   CA 3A
  CD   CA 4A
  CF   C2 3A   LOOP 19
  D1   02
  D2   EA 3B
  D4   CA 3A
  D6   C2 4A
  D8   EA 4B
  DA   CA 4A
  DC   C2 3C
  DE   02
  DF   EC 99
  E1   CA 3C
  E3   C2 4C
  E5   EC 99
  E7   CA 4C
  E9   9C E4   JNZ LOOP 19
  EB   C2 3C
  ED   9C E0   JNZ LOOP 19
  EF   90 82   JMP 0B73
  F1   C4 0F
  F3   37
  F4   C4 4F
  F6   33
  F7   3F      JMP 0F50
  F8   C4 0F
  FA   37
  FB   C4 75
  FD   33
  FE   3F      JMP 0F76
  FF   00      Spare

0F50   40
  51   E4 0A   FUNC is+?
  53   98 0D   JZ 0F62
  55   C4 9B
  57   02
  58   FA 3A
  5A   CA 3A
  5C   C4 99
  5E   FA 4A
  60   CA 4A
  62   C2 3A
  64   02
  65   EA 3B
0F67   CA 3A
  69   C2 4A
  6B   EA 4B
  6D   CA 4A
  6F   C4 0B
  71   37
  72   C4 72
  74   33
  75   3F      JMP 0B73
  76   C4 9B
  78   02
  79   FA 3A
  7B   CA 3C
  7D   C4 99
  7F   FA 4A
  81   CA 3C
  83   C4 00
  85   CA 3A
  87   CA 4A
  89   C2 3C   LOOP 20
  8B   02
  8C   EA 3B
  8E   CA 3B
  90   C2 4C
  92   EA 4B
  94   CA 4B
  96   06
  97   94 0F   JP if result neg
  99   C2 3A
  9B   02
  9C   EC 01
  9E   CA 3A
  A0   C2 4A
  A2   EC 00
  A4   CA 4A
  A6   90 E1   JMP LOOP 20
  A8   90 C5   JMP 0F6F
```

APPENDIX

Insert the following blocks to convert datafile from 2-figure to 4-figure numbers.

```
0442   C2 80   LD var L
  44   CA 17   ST TS1
  46   40
  47   02
  48   F4 10
  4A   01
  4B   C2 80

046B   C1 80   LD data H
  6D   CA 17   ST TS1
  6F   AA 1D   ILD DAK
  71   01
  72   C1 80   LD data L
  74   01
  75   C2 24
  77   02
```

ETI MAY 1980

THE BLACK HOLE

We proudly present the latest offering from Tim Orr, the prolific producer of music machines – the Black Hole Chorus Machine. It's capable of processing the output of both natural instruments and synthesisers.

In addition to the chorus effect you can also choose genuine vibrato. That's not all – you can select a 'double' chorus option. The speed of both effects can be controlled manually. If you're not into knob-twiddling or you don't have a free hand or two, the Black Hole can be controlled by footswitch. Keep up with what's happening in music machines and much, much more in the next audio special issue of ETI.

KIT SURVEY

Across the length and breadth of this sceptred isle, there are companies producing kits of everything from power supplies and pin ball games to amplifiers and ignition systems. Want to buy a kit? How do you know who the supplier is, where he is, how reliable his product is and how much it costs? You could search through a dozen or so electronics magazines and spend a small fortune on postage to collect a library of catalogues.

Why don't you do it the easy way? Let ETI's fingers do the walking for you. Next month we get it all together – kits, suppliers, prices, quality – in an easy to compare format.

IMAGE CO-ORDINATOR

How to throw your voice without straining your vitals – build the ETI Image Co-ordinator. The clever co-ordinator takes your single vocal (or guitar, etc.) input and splits it in two. What can you do with two half voices? You can recreate a single sound image and make it move around, suggesting a few interesting stage and studio effects. The Image Co-ordinator uses two of the 1537A VCA chips introduced by Keith Brindley in March.

LED VU

Banish the bearings from your VU meters. Change over to a stylish LED display. Our LED VU meter is based on the LM3915, a chip which gives you VU or peak programme (PPM) options with bar or dot display. Look in next month to see the VU from ETI.

SERVO TESTER

Last month's Radio Control Fail–Safe stops your plane or boat disappearing into the sunset if you lose control of a channel, for whatever reason. When you get your plane or boat back onto dry land, a thorough systems check is number one on the list of things to do. A servo fails to operate. Is it the servo or the receiver? You can eliminate the servo by using our servo tester – an unusual and useful little piece of test gear.

SYNTHESISER

The Project 80 Modular Synthesiser returns with designs for the four filters most widely used in music synthesis – low pass, high pass, band pass and phase shift. They are four pole filters with one volt per octave control of their cut-off, or centre, frequency. Voltage control of signal regeneration is also included.

Ian Sinclair.

In our new course we take a look at the guts of your MPU before expounding its capabilities.

The message program in last month's piece was just the start of the vast range of 'data processing' uses, as distinct from arithmetic, which we can carry out using the MPU. We're going to carry on in that vein for some way yet, because these are the real nitty-gritty of what an MPU is designed to do. The first subject this week is Pattern Recognition, and the program is shown in Fig.1.

```
0F13   C5   LD@P1
0F14   01   01
0F15   E4   XRI
0F16   0A   Byte
0F17   9C   JNZ
0F18   FA   to 0F13 again
0F19   3F   return to monitor.
```

To set up: ABORT : 0FF9 ; Term ; 0F ; MEM ; 1A ; ABORT ; 0F13 ; GO
To read answer: ABORT; 0FFA gives lower byte of address +1
Example: if 0FFA shows 30, the byte we're looking for is in 0F30 – 1 = 0F2F.

Fig.1. The pattern recognition program. This one searches memory for a byte which it has to recognise. The program stops when the byte is found.

Bit Search

The basic idea is very simple. When you switch on the Mk14, the RAM will store a 'rubbish' byte at every address. Some of these seem to turn up more than others, but in general they seem to be at random. The program examines each bit, starting at 0F1A, and stops when a particular byte is found. The byte we are looking for has to be entered into the program at 0F16 – in Fig.1 we've selected 0A. If there isn't an 0A stored anywhere between 0F1A and 0FF8, then the program will stop and show the address 0F17, because the program has been right round, and the first 0A it has found is in itself! If there is an 0A in memory, the program stops at one step beyond it. For example, if there's 0A at 0F2F, then the program will stop at 0F30 – you can abort and address 0F2F to make sure!.

What does the program do? To start off with, pointer P1 has to be loaded up with the starting address, which in this case is 0F1A. The first instruction of the program is an indexed load relative to P1, so that the byte from 0F1A is placed in the accumulator. The next step is X–OR'd with itself, the result is always zero, Fig.2, so that if the byte 2A existed at address 0F1A, then the result of the X–OR step would be zero. The step at 0F17, 18 is a jump-if-not-zero, so that if the byte 0A has been found, the program goes to 0F19, which is return-to-monitor, 3F. If, however, the byte fetched in from 0F1A was not 2A, then the accumulator is not at zero after the X–OR step, and the jump takes place. The jump is back to 0F13, the start of the program. Because of the auto indexing of the step at 0F13, the next byte which is fetched will be from 0F1B, and the comparison is made again.

X–OR LAWS

$\oplus$ indicates X–OR

$0 \oplus 0 = 0$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

Fig.2. The X–OR action – a reminder.

Chunky Stuff?

Another useful chunk of program which follows directly from the work we did in Part 9 is a memory block shift. The aim here is to take a number of bytes from one place in memory and copy them to some other place. It's the same sort of action as the 'message' program, and is detailed in Fig.3. The program starts at 0F1F, which is used to store the number of bytes which are to be shifted. Pointer P1 is then set up with the starting address of the memory block which is to be shifted, and P2 is loaded with the starting address of the new block. The example shows ten bytes (0A at 0F1F) shifted from a starting address at 0F30 to a new starting address at 0F50. These addresses have to be loaded into the pointer registers in the usual way, placing 0F at 0FF0, 0FFB, 30 at 0FFA and 50 at 0FFC.

The program uses the auto indexed load (relative to P1) to place a byte from 0F30 into the accumulator, and then the auto-indexed store (relative to P2) places that byte into 0F30. The number-of-bytes figure stored at 0F1F is then decremented and loaded into the accumulator, and followed by a jump-if-not-zero instruction. The jump is back to the load instruction, which because of the auto indexing is from 0F31 and is followed by a store to 0F51. This continues until the figure in 0F1F is 01. When this is decremented and loaded, the result in the accumulator is 00, so that there is no jump and the program finishes with a return to the monitor.

Use of registers: P1 – start of old block ; P2 start of new block.

```
0F1F   0A   Number of bytes to shift (NOBT)
0F20   C5   LD@P1 OLD
0F21   01   01
0F22   CE   ST@P2 NEW
0F23   01   01
0F24   B8   DLD NOBT
0F25   FA   NOBT
0F26   9C   JNZ
0F27   F8   OLD
0F28   3F   Return to monitor.
```

To set up: 0FF9 ; Term ; 0F ; Mem ; 30 ; Mem ; 0F ; Mem ; 50 ; ABORT ; 0F20 ; GO
At end of run: ABORT ; 0F30 – note data bytes for ten bytes on and then ABORT ; 0F50 – note bytes from here on; they should be identical with the bytes shifted from 0F30 on.

Fig.3. The Memory-block shift program. Pointer registers P1 and P2 are used to contain the starting addresses of the two memory blocks.

By The Left. . .

Now for something which incorporates last month's work with what we've done so far. It's not a simple program by any means, and what makes it interesting at this stage is that the S of C manual achieves the same effect by a rather different method. The idea is to make a moving message — writing a message on the LED's and shifting all the letters one place to the left at intervals.

LED	0F1D	00
DLY	0F1E	FF
OPT	0F1F	00
	0F20	C4
	0F21	08
	0F22	C8
	0F23	FC
	0F24	C6
	0F25	01
	0F26	CD
	0F27	01
	0F28	B8
	0F29	F6
	0F2A	9C
	0F2B	F8
	0F2C	C0
	0F2D	F0
	0F2E	31
	0F2F	C4
	0F30	50
	0F31	32
	0F32	B8
	0F33	EB
	0F34	9C
	0F35	EA
	0F36	C4
	0F37	FF
	0F38	C8
	0F39	EA
	0F3A	A8
	0F3B	E2
	0F3C	E4
	0F3D	08
	0F3E	9C
	0F3F	E0
	0F40	C4
	0F41	00
	0F42	C8
	0F43	DA
	0F44	90
	0F45	DA
	0F50	33
	0F51	30
	0F52	5F
	0F53	38
	0F54	39
	0F55	37
	0F56	30
	0F57	6D
Setting up :		
	0FF9	0D
	0FFA	00
	0FFB	0F
	0FFC	50

Fig.4. Moving message program. This one is a logical development of the static message program used earlier.

The method uses the same basic message program as we developed in Part 9, but with several important changes. In the simple message program, a byte was loaded, using auto indexing relative to P2 and stored auto indexed relative to P1. After a count of eight LED's, the original addresses were restored in the pointers so that the same messages could be run again.

This time we don't want the message go start at the same LED address (0D00) each time. We want to run one lot starting at 0D00, keep it going for a time, then start from 0D01 (the next LED along), keep this one on for a time, then start at 0D02 and so on until the message has disappeared off the end of the display — then we want to start from scratch again.

The program is shown in Fig.4. Because it's fairly long, extra care is needed to check it, because if it crashes, you'll probably find corruption — the contents of the RAM will have been written over so that parts of the program have been changed into gibberish. In part 12 we'll deal with debugging and how to cope with such difficulties when you're writing your own programs. For the moment, let's go through this one and see what it does.

The program has various numbers stored at 0F1D, 0F1E, 0F1F. At 0F1E we've stored FF. This is a delay byte which affects how quickly or slowly the message moves, and you can experiment with changing this value once you have the program running.

The action starts at 0F20, 21 where 08 is loaded. This is just the LED count which we used before, it ensures that we switch on the eight LED's we are going to use in turn. We then store this number back at address 0F1F by using the C8FC steps. Remember that the number stored in 0F1F will be decremented on each run, and we need to be able to re-load it; this lot so far is the re-load part of the program.

The steps 0F24 to 0F27 should now be familiar — C601 takes a byte from a place in RAM auto indexed by P2 and then CD01 stores it auto indexed to P1. P2 is set to start at the point in RAM where our message begins, and P1 is set to point to 0D00, the first LED on the right hand side. Because auto indexing is used, each of these addresses will increment on each run.

Having dispatched one pattern to one LED, we have to attend to the next one and the next four instructions B8 F6 9C F8 do just that, decrementing the count (stored at 0F1F) and jumping back to address 0F24 to load up another byte and display it.

So far, I've been reminding you of established steps, but from now on we're plunging into the unknown, so fasten your belts. The next two steps at 0F2C, 0F2D, load in the number from address 0F1D. This happens only when the message has been displayed for one run around the LED, because it follows the jump-if-not-zero instruction at 0F2A. In the message program of Part 9, we used a load immediate here — it serves to return pointer P1 low byte to its starting address. For a static message, the starting address is always 0D00, but for a moving message program in the manual, and it is, I hope, a bit easier to follow.

Having got this byte, which will be 00 for starters, we slap it into pointer P1 by using the 31 instruction. So far, so good. Next step (starting at 0F2F) is to load immediate 50 and exchange with the low byte of P2. That restores the 0F50 address in P2 so that we can start again with the data; pretty much the same as the static message program.

A Quick Flash

So far, the steps of the program will flash a message up, but not long enough to see. We need a bit of time to look at what is displayed, and so the next step provides a bit of delay by making the program from 0F24 to 0F33 loop round. At 0F32 we have B8, decrement and load, followed by E8, which fetches from address 0F1E. Now we started with a fairly large number stored in 0F1E, so that there will be that many loops around this first part of the program. Why didn't we use the DLY instruction, do I hear you ask? Well, it would provide a delay all right, but when you put DLY in, the program sticks there, and the display goes blank! Not what we want at all, so we must use the loop method – the delay is quite long enough without doing anything fancy like running two delay steps in series. At 0F34, the 9C instruction is a jump-if-not-zero and 0F35 ensures that the jump is to the start at 0F20, so that a complete sweep of the LED's is done on each loop, giving us a static message for the duration of the loop while the number in 0F1E is counted down. At the end of the loop, there's no jump back and we're into new territory at 0F36.

What we want to do now is to shift the starting point of the message and run it again for the same time. Obviously we'll need to reload the delay figure in 0F1E, so the next set of steps, 0F36 to 0F39, does just that. A load immediate is used, and FF ensures the maximum delay. If you want to change the delay, then this is the byte to play with – try 80 or even less if your eyes can follow the speed of movement. The C8 EA bytes then load FF into address 0F1E ready for the next run.

Now we want to have our next run with the beginning of the message on the second LED from the right. Remember that messages are entered from the right so 'beginning' in this sense means the first byte from memory as far as reading the message is concerned, it's actually the end!

At 0F3A, then we have an *increment and load* instruction A8, referred to address 0F1D. This, you may remember, is the address we use for loading the low byte into pointer P1 for starting the display, so this is the important step for causing the message to move. We can't leave it at that, though, because address 0F1D will be incremented until the cows come home unless we do something to stop it. Since we don't have more than 8 or 9 LED's operating, and we need only 8 for this message, we'll stop it at eight. a spot of pattern recognition then follows, using XR1 8 (address 0F3C, 0F3D) to see if the LED start address has got to 8. If it hasn't, the result of the XOR will leave a byte in the accumulator. We're not interested in the size of the byte, just that it isn't zero until the LED number is 8, so we can use a JNZ at 0F3E.

Now where does this take us? Well, if we haven't reached LED 8 (counting from the right hand side), we jumped back to address 0F20 to start another complete display run. That way we're going to display our message starting at each LED in turn from the first on the right hand side, showing each one long enough to see, then jumping to the next. At the end of this carry on, of course, there's only one LED left carrying the message! The rest of the message is being delivered to addresses which don't exist unless you've tacked on another display. At this point, the count in 0F1D reaches 8, the XR1 step results in a zero at step 0F3D, and the JNZ lets the program step onto 0F40.

We've now done a sweep of moving message, and it only remains to reset everything and start again in an endless loop to keep it all going. At 0F40 we load immediate 00 to restore the P1 pointer by storing this byte back into 0F1D (step 0F42, 0F43). We then jump back into the endless loop with the 90, DA bytes at 0F44,45 which take us back to the starting address of 0F20.

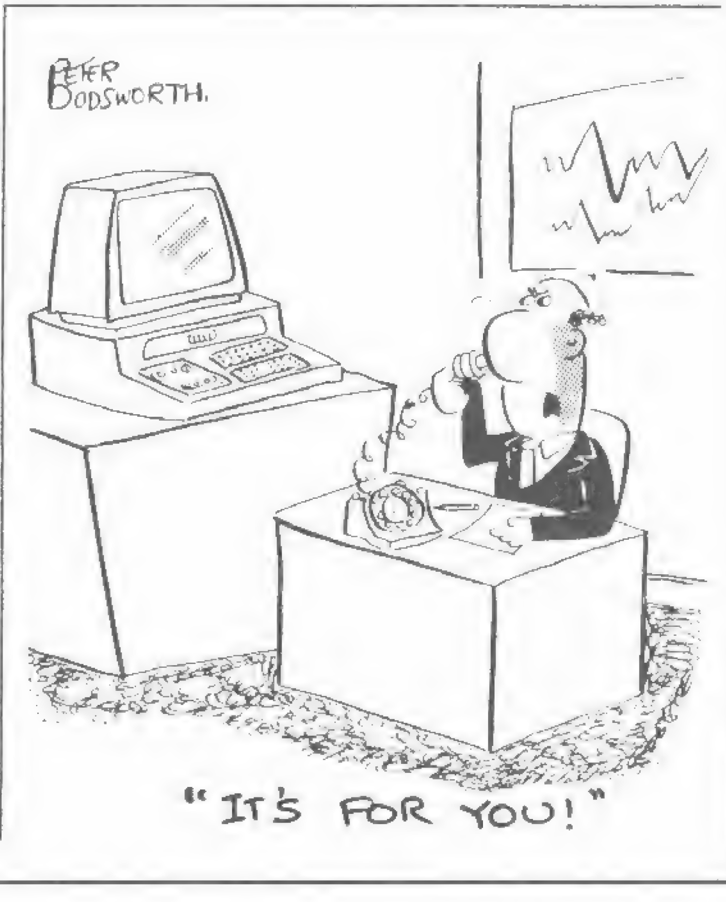
New Table :

0F50	33
0F51	30
0F52	5F
0F53	38
0F54	39
0F55	37
0F56	30
0F57	6D
0F58	30
0F59	5F
0F5A	38
0F5B	39
0F5C	37
0F5D	30
0F5E	6D

Fig.5. Extended table for the moving message program. The rest of the program can remain unchanged.

The message has to be loaded into 0F50 to 0F57 as before, last letter first. The pointers have to be set up by selecting 0FF9 and loading in 0D then stepping them to load 00, 0F, 50. Only then can we abort (NOT reset – that'll put all the pointers back to zero), address 0F20 and GO.

All very well, I hear you say, but it's not a *real* moving message. It's a message and it moves, that's all. If that's all you want, no problem. Just use the data table shown in Fig.5, and change the number at 0F21 to 10 (decimal sixteen). Now run it, and see the difference. You can spend the rest of the month thinking about that one!



FLETCH


ROVA/PRO

Dear Sir,

The program given on page 36 of the January issue, for the binary search of an ordered list has one fundamental fault. It does not detect that an item T is not in the list, except by not responding in a finite time. This is due to the action of the INT function in line 130, as mentioned in the text, a result of which is that L and U can never be equal. The simplest change is to amend line 150 thus :—

150 IF L=U–1 THEN 210

However an alternative approach is possible which removes the special case of T=A(U) in line 120. When T is not at the current position S, then the position S need not be in the new range of the list which includes T (if there). Thus the new values of U and L can be U=S–1 and L=S+1 depending on the half range rejected. It is now possible for L and U to cross necessitating a change to 150. In this case the changes to the program are :—

i) delete lines 120,230

ii) rewrite the following lines as shown

```
150 IF L >=U THEN 210
170 U=S–1
190 L=S+1
```

Yours etc.,
C. Hayward.

North Cheshire College,
Fearnhead,
Warrington WA2 0DB.

Dear Sir,

There doesn't seem to be any other Social Worker around who is actually using a Micro in his daily work. It seems rather surprisingly thin ground, in view of the multifarious applications I find for mine — from writing rude memo's to the Town Hall, through doing the Unit accounts, to beginning to look at some very 'micro' research.

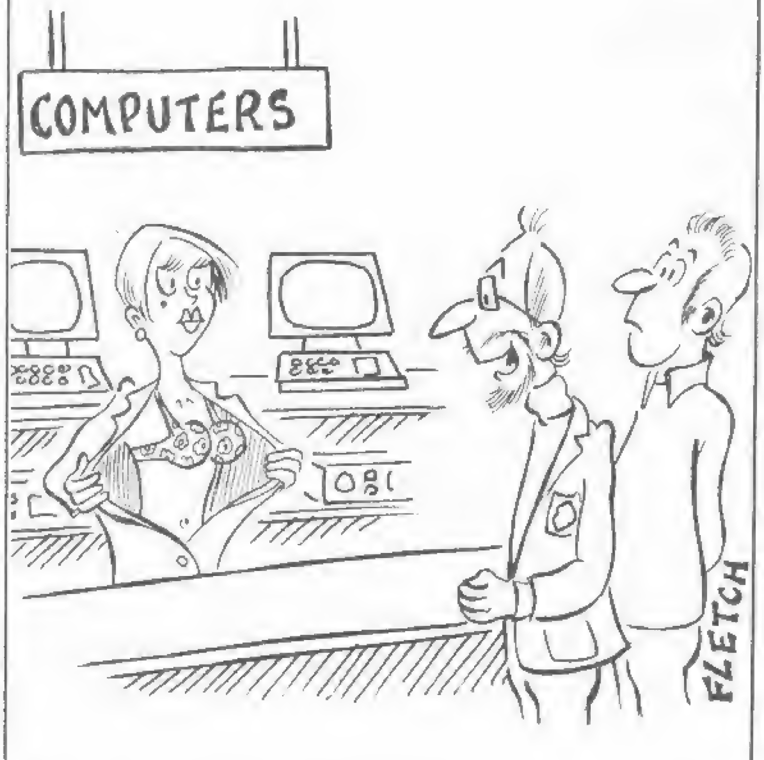
Is there anyone else out there ?

Incidentally, this letter is written on a TRS-80 using a modified 'Electric Pencil', typed on the DTN mod of the S/H IBM Golfball, and filed on disk. Beats using my clerical officer hollow (though please don't tell her that!).

Yours, etc.,
John Wallbridge.

17 Granville Road,
London N.4.

P.S. My specialised program for psuedo — randomly swearing at Local Government Departments (in Extra-Basic), together with an on-going social-work and psychological jargon generator is available to anyone in need of this special therapy!



"I JUST ASKED IF SHE COULD SHOW ME ANY SOFTWARE..."

Dear Sir,

I would like to point out, with reference to the mention of the TRS-80's keyboard bounce in Ian Sinclair's article, page 20 in the February issue, that the keyboard is easily "de-bounced".

To do this, first remove the keyboard fascia, by prising it up with a screwdriver blade, and then pull off the plastic key cap, starting with the shift keys and working inward but do not try to remove the SPACE key (Some of the keys may need quite a strong pull but by working it off slowly you should be able to remove all of them.) Then clean each of the contacts by inserting a piece of stiff paper between the contacts, pressing down the key, and pulling out the paper while the contacts are still pinching it. If you want to do a really thorough job you can get some contact cleaner (Tandy sell it for about a pound) and spray it into the contact. You then replace the keys (In the right order!) and put the fascia back on. Having done this there should be no more trouble with keyboard bounce.

Yours sincerely,
Tim Adye.

The Abbey Old House,
Cowl Lane, Winchcombe,
Glos. GL54 5RA.

P.S. Although keys cannot be removed individually, Mr. Sinclair will find that he can get them off and clean the contacts of his TRS-80 by the above procedure.

Dear Mr. Lusty,

I hope you do not mind correspondence on your column in Computing Today. I was interested in the square triangle problem you set, so felt duty bound to write to you about it.

Did you intend to send people off in the wrong direction when you asked them to ensure that their square root routines worked correctly? I do not believe it is necessary to use square roots in this problem and suggest an alternative which should surely work quicker.

I await your rebuttal.

My TRITON Tiny Basic solution as follows :

```
10    X=1;  Y=1;  D=0
20    PRINT X, Y;  GOSUB 30;  GOTO 20
30    D=D+Y+Y+1;  Y=Y+1
40    IF D > 0  X=X+1;  D=D-X;  GOTO 40
50    IF D=0 RETURN
60    GOTO 30
```

On the TRITON of course this will not give all the values up to 1,000,000 but it will on any machine capable of working with them.

There is no need for the code in lines 30 thru 60 to be a subroutine it just made it slightly more easy to replace these lines by a machine code subroutine. This little exercise showed up the great disparity in times taken between the Tiny BASIC interpreter and the raw code.

I leave it to you to unravel the reasoning in the above program.

Yours sincerely,
John Senior,

20, Great Ley,
Welwyn Garden City,
Hertfordshire.

Dear Sir,

N2 Review — February Issue CT

As an owner of the above machine I read your review with great interest — it is always interesting to read about one's own equipment. My only argument with the article would be with the benchmark test figures quoted. In fact the N2 BASIC chip will not work at 4 MHz without the wait state — the notes supplied by Henry's Radio confirm that it is not just my machine. I therefore assume you were given a 'souped up' machine or else were supplied with the figures and didn't check.

Further the BASIC commands SET, RESET and POINT will only work properly if one has the extra graphics ROM, but it is well worth it. Another worthwhile extra is the Port Probe sold by Bits and PCs (18 Rye Garth, Wetherby, West Yorks), I have found it an excellant device for learning how to use the PIO and it gives the INP, OUT and WAIT commands something to do.

Yours faithfully,
Dr. C.V. Nowikow.

144 East Park,
Harlow,
Essex, CM17 0SA.

Dear Editor,

There is a catastrophic misprint in the Logic Emulator in your February issue, which prevents the program from working at all : location D52 should read 0A not A0.

There is also a misprint in Message 2 : location E24 should read 4F not 45.

Yours sincerely,
T.P. Goldingham.

Wyndham,
11 Furze Platt Road,
Maidenhead,
Berkshire. SL6 7ND.

Trevor Lusty replies :—

No, Mr. Senior, I had not intended to send you in the wrong direction. I felt that many people would attempt to solve the problem using the square root function and that they should first check its accuracy. I chose to solve the problem in this way to highlight and explain the potential difficulties.

That said, may I congratulate you on an excellent solution. Don't underestimate it ——— as you work with the sides of the triangles and squares the number of balls exceed 1,000,000 as soon as Y exceeds 1000.

For anyone who finds Mr. Senior's solution difficult to follow, here is a guide. The solution depends on two mathematical facts :—

1) The n th square number is the sum of the first n odd numbers.

eg. $6^2 = 1 + 3 + 5 + 7 + 9 + 11$

2) The n th triangle number is the sum of the first n integers.

eg. 8 th triangle number = 1+2+3+4+5+6+7+8

Mr. Senior's solution is particularly clever in the way in which store D is used to hold the difference between the partial sums of these two series. The following program uses the same method as Mr. Senior but keeps the sums of the series in different locations, the squares in store S and the triangle numbers in store T, and this, I believe, makes it easier to understand the method.

```
10    LET X=1: LET Y=1: LET S=1: LET T=1
20    IF S < T THEN Y=Y+2: S=S+Y: GOTO 20
30    IF S > T THEN X=X+1: T=T+X: GOTO 30
40    IF S <> T THEN 20
50    PRINT "TRI";X,"SQR";SQR(S),"TOTAL";S
60    LET Y=Y+2: S=S+Y: GOTO 30
TRI      1    SQR     1   TOTAL        1
TRI      8    SQR     6   TOTAL       36
TRI     49    SQR    35   TOTAL     1225
TRI    288    SQR   204   TOTAL    41616
TRI   1681    SQR  1189   TOTAL  1413721
```

As Mr. Senior is impressed by the machine code version of his program, he might like to compare it to the following BASIC program. It is the fastest solution I have found to the Square Triangles problem ——— I shall return the compliment and let Mr. Senior unravel the reasoning.

```
10    REM *** SQUARE TRIANGLES ***
20    REM ***    FAST SOLUTION    ***
30    LET S1=0  :  S=1  :  T1=0  :  T=1
40    PRINT  :  PRINT
50    PRINT  "SIDE OF TRIANGLE IS" ; T
60    PRINT  "SIDE OF SQUARE IS" ; S
70    PRINT  "TOTAL NO. OF BALLS IS" ; S*S
80    LET T2=T  :  S2=S  :  T=6*T-T1+2
90    LET S=6*S-S1  :  T1=T2  :  S1=S2
100   IF T < 2000 THEN 40
110   END
```

lete computer kit.

£79.95
Including VAT.
Including post and packing.
Including all leads and components

Fewer chips, compact design, volume production – more power per pound!

The ZX80 owes its remarkable low price to its remarkable design: the whole system is packed onto fewer, newer, more powerful and advanced LSI chips. A single SUPER ROM, for instance, contains the BASIC interpreter, the character set, operating system, and monitor. And the ZX80's 1K byte RAM is roughly equivalent to 4K bytes in a conventional computer, because the ZX80's brilliant design packs the RAM so much more tightly. (Key words, for instance, occupy just a single byte.)

To all that, add volume production – and you've that rare thing: a price breakthrough that really *is* a breakthrough.

The Sinclair ZX80. Kit: £79.95. Assembled: £99.95. Complete!

The ZX80 kit costs a mere £79.95. Can't wait to have a ZX80 up and running? No problem! It's also available, ready assembled, for only £99.95.

Whether you choose the kit or the ready-made, you can be sure of world-famous Sinclair technology – and years of satisfying use. (Science of Cambridge Ltd is one of the Sinclair companies owned and run by Clive Sinclair.)

To order, complete the coupon, and post to Science of Cambridge for delivery within 28 days. Return as received within 14 days for full money refund if not completely satisfied.

sinclair ZX80

Science of Cambridge Ltd
6 Kings Parade, Cambridge, Cambs., CB2 1SN
Tel: 0223 311488.

Order Form

To: Science of Cambridge Ltd, 6 Kings Parade, Cambridge, Cambs., CB2 1SN.
Remember: all prices shown *include* VAT, postage and packing. No hidden extras.

Please send me:

Quantity	Item	Item price £	Total £
	Sinclair ZX80 Personal Computer kit(s). Price includes ZX80 BASIC manual, excludes mains adaptor.	79.95	
	Ready-assembled Sinclair ZX80 Personal Computer(s). Price includes ZX80 BASIC manual, excludes mains adaptor.	99.95	
	Mains Adaptor(s) (600 mA at 9 V DC nominal unregulated).	8.95	
	Memory Expansion Board(s) (takes up to 3K bytes).	12.00	
	RAM Memory chips – standard 1K bytes capacity	16.00	
	Sinclair ZX80 Manual(s) (manual free with every ZX80 kit or ready-made computer).	5.00	
NB. Your Sinclair ZX80 may qualify as a business expense.		TOTAL	£

I enclose a cheque/postal order payable to Science of Cambridge Ltd for £ ....
Please print
Name: Mr/Mrs/Miss ....
Address ....

CT 4/80

Dr. G.J. Marshall.

Dr Marshall, author of 'Principles of Data Communication' reveals the facts on the most popular choices.

In his book 'Programming Languages: history and fundamentals', Jean Sammett mentions well over a hundred languages, all of which were in use in the USA at the time. Since the publication of the book, in 1969, even more languages have been invented, particularly for applications such as real-time control and automatic testing. Clearly, to attempt to review *all* programming languages is a daunting task. To review the most popular, or widely used, ones is considerably easier, for many a language has never been used by anyone except its inventor and his immediate circle.

The first really popular high-level language for scientific applications was FORTRAN, which emerged from IBM in 1957. Among languages of the same type, its main competitors have been ALGOL 60 and BASIC. The latter has gained considerably in popularity of late as a result of its wide availability on microcomputers. For commercial data processing, COBOL is even more popular than FORTRAN is in its field, having overwhelmed practically all competition.

An increasing awareness of the shortcomings of these languages lead to the development of the 'super languages' ALGOL 68 and PL/1. These languages were designed, in their contrasting ways, to be suitable for both scientific and commercial programming. Additionally, both provide the facilities necessary for structured programming, since they possess the control features necessary to write large programs in a modular fashion. This simplifies the writing of large programs and also the task of maintaining them.

However, the 'super languages' proved difficult to implement satisfactorily, and the available implementations occupy large amounts of store. The desire to retain the features of these languages, but in more compact implementations, has lead to the development of new languages that seem likely to be the real popular successors of the first scientific and business languages. Thus, ALGOL 68 has spawned Pascal, and PL/1 has lead to PLM and a group of similar languages. These new languages are well suited for use with microcomputers.

Outside this main stream of development, many languages have been developed for specialised applications. Among the most popular special purpose languages are LISP, for list processing, SNOBOL, for string handling, and PILOT, a small language for text processing. Languages for real-time applications include CORAL 66, while ATLAS is a language for automatic testing.

The Survey

A selection of the most popular programming languages is surveyed by describing the major features of each in a way that illustrates its capabilities. By and large, a programming language should possess, at the least, facilities for computing and storing values, input and output, conditional execution of instructions, repetition, sub-programs and data structures. It is useful to have a set of test problems to illustrate the different methods of solution that are permitted by, or imposed by, different languages. The following test problems are used :

Problem 1. Program the formula $x = a + \frac{2b}{c}$

Problem 2. Program the formula $x = \sqrt{\log_e \sin(a)}$

Problem 3. Print out the integers from 10 to 20

Problem 4. Accept and store a set of numbers, find the largest and print it out.

Problem 5. Store a set of English words and the corresponding French words. Then accept an English word and print out its French equivalent, thus appearing to translate.

These problems are mainly numerical, reflecting the bias in the language selection. However, Problem 5 gives some idea of the suitability of a language for data processing, requiring as it does the structuring of data and a search procedure.

FORTRAN

FORTRAN owes its supremacy over the early high-level scientific programming languages to its support from IBM. Once established as the language that most scientific programmers knew and in which most scientific software was written, it was naturally difficult to dislodge. Although there are many dialects of FORTRAN, the definitive version is ANSI standard FORTRAN IV.

The way in which arithmetic computation and the storage of values is achieved in FORTRAN is illustrated by the following instruction which solves Problem 1 :

```
X = A + 2.0*B/C
```

The effect of this instruction is to cause the expression to the right of the equals sign to be evaluated, and to assign the resulting value to the variable X. In the expression, the multiplication must be indicated explicitly (by the star). Since the variables A, B and C are, by implication, real valued, the two is written as the real number 2.0 rather than as the integer 2 to avoid mixed-mode arithmetic. FORTRAN automatically performs arithmetic operations in the correct order, so that, for instance, multiplications are performed before additions. Brackets can be used to change this order in exactly the same way as in algebraic formulae. Thus the effect of the program segment

```
A = 6.0
B = 5.0
C = 4.0
X = A + 2.0*B/C
```

is that the value 8.5 is assigned to X.

Problem 2 is solved quite simply in FORTRAN which possesses a range of standard functions broadly comparable to that of a scientific calculator. The solution is

```
X = SQRT (ALOG (SIN (A)))
```

The value of A is treated as a number of radians by the sine function.

Input and output are achieved with READ and WRITE instructions, thus a program segment to read a number from an input device and to write it to an output device immediately is :

```
      READ (1, 100) A
100   FORMAT (F10.3)
      WRITE (2,101) A
101   FORMAT (F11.3)
```

In each READ or WRITE instruction, the key word is followed by a pair of numbers in brackets. The first number is a device number, so that in this example 1 is the number of an input device – a card reader, say – and 2 is an output device number – of a line printer, perhaps. The second number gives the label of the associated format statement which every input/output instruction must have. The necessity of formats can be aggravating, but it gives the programmers complete control over the layout of his input and output. In the example, the value of A is specified as a floating point value (F), punched in the first 10 columns of a card and having 3 figures after the decimal point. The output format similarly specifies the way in which the line printer should print the value of A.

Conditional instructions have the form :

IF (condition) instruction

The condition involves the comparison of two values, and thefollowing instruction is executed only if the condition is true. Otherwise control passes to the next instruction. A typical conditional instruction is

```
IF (A.EQ.6.5) X = A + B
```

and when this instruction is executed the sum of A and B is assigned to X only if the most recent value assigned to A is 6.5.

Repetition is achieved with a DO loop. This facility gives the automatic repetition of all the instructions between a DO and its matching CONTINUE statement as often as indicated. Thus Problem 3 is solved by

```
      DO 50 I = 1,11
      J = I + 9
      WRITE (2,100) J
100   FORMAT (I3)
 50   CONTINUE
```

The number following DO is the label of the matching CONTINUE. The variable I is the loop counter. It counts the repetitions, and here repetition starts with I set to one and continues while I increases by one until it reaches 11.

FORTRAN supports both functions and subroutines. The function sub-program computes a single value and returns it to the main program. A subroutine can return multiple values besides being executed for its side effects.

The only data structure available in FORTRAN is the array. The declaration statement

```
DIMENSION A(50)
```

reserves storage space for a one dimensional array, A, with elements A(1) to A(50), each of which can be manipulated in exactly the same way as an ordinary variable. A program for Problem 4 that deals with a set of 10 numbers is :

```
      DIMENSION A(10)
      READ (1,100) A(1)
100   FORMAT (F10.2)
      AMAX = A(1)
      DO 15 I = 1,9
      J = I + 1
      READ (1,100) A(J)
      IF (A(J).GT.AMAX) AMAX = A(J)
 15   CONTINUE
      WRITE (2,101) AMAX
101   FORMAT (F11.2)
```

After execution of this segment the 10 numbers are stored in the array, A, and AMAX has been assigned the value of the largest.

The handling of strings and characters in FORTRAN is somewhat limited, so a solution to Problem 5 is not presented. However, some dialects permit a solution similar to the one presented in the section on BASIC.

BASIC

BASIC was devised at Dartmouth College in the USA as a high-level language that would be easy to learn and to teach. Its recent rapid increase in popularity has stemmed from the speed with which it can be learnt and from its ready availability on microcomputers. It is the language that is available on the Commodore PET and the APPLE. Although there is a standard version of BASIC, so many variations and extensions are currently available, including extensions for text processing or real-time applications, that the standard has little meaning.

Instructions for computation and storage are almost identical to those in FORTRAN. The BASIC for Problem 1 is

```
LET X = A + 2*B/C
```

The LET is usually dropped. The instruction for Problem 2 is

```
X = SQR(LOG (SIN (A)))
```

A number can be input and immediately output by

```
10 INPUT A
20 PRINT A
```

The input instruction is interactive, and when executed causes the machine to wait until an input is entered from the keyboard. In BASIC programs, every instruction has a line number. Before executing a program BASIC uses the line numbers to sort the instructions into order. There is also a READ instruction that reads from DATA statements included in the same program.

Conditional instructions have the form

IF condition THEN instruction

for example

```
IF A > 6.5 THEN X = A +B
```

Their execution is similar to that of conditionals in FORTRAN.

For repetition, the key words to start and end a loop are FOR and NEXT. A program for Problem 3 is

```
10   FOR I = 10 TO 20
20   PRINT I
30   NEXT I
```

Arrays are supported by BASIC, although the declaration

DIM A(20)

reserves space for the one-dimensional array with elements A(0) to A(20). A program for Problem 4 that deals with a set of 10 numbers is :

```
10  DIM A(10)
20  INPUT A(1)
30  AM = A(1)
40  FOR I = 2 TO 10
50  INPUT A(I)
60  IF A(I) > AM THEN AM = A(I)
70  NEXT I
80  PRINT AM
```

BASIC provides facilities for handling strings. A variable whose name ends in $ can have a character string assigned to it. This transparent, but not very efficient, program provides a solution to Problem 5 for a vocabulary of 21 words :

```
 10  DIM E$(20), F$(20)
 20  E$(0) = "HOUSE"
 30  F$(0) = "MAISON"
 40  E$(1) = "CHAIR"
 50  F$ (1) = "CHAISE"
   etc.
200  INPUT "ENTER ENGLISH WORD", A$
210  B = 0
220  FOR I = 0 TO 20
230  IF A$ = E$(I) THEN PRINT F$(I)
240  IF A$ = E$(I) THEN B = 1
250  NEXT I
260  IF B = 0 THEN PRINT A$; "NOT IN
     VOCABULARY"
270  GOTO 200
```

ALGOL

ALGOL 60 is formally defined in a report dated 1960, and although it is a more rational language than FORTRAN, it has never managed to dent the popularity of the latter to any marked degree.

Its computation and assignment instructions are typified by the instruction

```
x:= a + 2.0*b/c
```

The avoidance of a simple equals sign reminds the programmer of the assignment required by this instruction.

Input/output is the one language feature not defined in the ALGOL 60 report, so that it varies from implementation to implementation. A value is read and printed out in ICL ALGOL 60 by

```
a:= read ; print (a,3,2)
```

The semicolon acts as an instruction separator. The print statement delivers the value of a with three places before the decimal point and two after it.

A typical conditional instruction is

```
if a > 6.5 then x:= a + b else x:= a – b
```

The repetition facilities can be illustrated by the following program for Problem 3.

```
for i:= 10 step 1 until 20 do
begin print (i, 2, 0) end
```

Both conditional and repetition instructions are quite explicit.

Arrays are supported, and their use is illustrated by the following program for Problem 4.

```
real array a(1:10) ; real amax;
a(1) := read ; amax := a(1);
for i:= 2 step 1 until 10 do
begin a(i):= read; if a(i) > amax then amax:= a(i) end;
print (amax, 3, 3)
```

All variables must be declared before they are used in ALGOL programs.

The sub-program in ALGOL in the procedure. Unlike FORTRAN and BASIC, ALGOL supports recursion, that is, sub-programs may call themselves.

COBOL

The pre-eminence of COBOL for business data processing stems from the US Government policy that required the provision of a COBOL compiler with any computer bought using their funding. As a commercial language, COBOL emphasises the handling of alphanumeric data and files, so that tasks such as reading and updating file records and automatic form filling can be accomplished.

The language is intended to be readable, having instructions such as

MOVE X TO Y

that cause single values or complete structures to be moved. Only simple arithmetic facilities are required; a typical instruction is

ADD BALANCE TO OLDTOTAL GIVING NEWTOTAL.

Problem 1 can be solved by

```
DIVIDE C INTO B.
MULTIPLY 2 BY B.
ADD B TO A GIVING X.
```

Here, the programmer must order the arithmetic operations. An alternative solution is

```
COMPUTE X = A + 2 * B / C
```

COBOL does not possess, or need, facilities to solve problems like Problem 2. It has READ and WRITE instructions for input and output, and conditional instructions such as

```
IF ORDER IS GREATER THAN 100
MULTIPLY DISCOUNT BY PRICE.
```

COBOL programs contain separate data divisions and procedure divisions. A file called CARDS with records called REC each of which consists of a single number, QUANTITY, of up to four digits can be declared in the data division by

```
FD    CARDS
      DATA RECORD IS REC.
01    REC.
      02 QUANTITY PICTURE 9999.
```

Thus, each record in the file contains a single number. The largest number in the file can be found, in the procedure division, in this way :

```
      MOVE ZEROS TO A.
READ–IN
      READ CARDS AT END GO TO LABEL.
      IF QUANTITY IS GREATER THAN A
      MOVE QUANTITY TO A.
      GO TO READ–IN.
LABEL.
      WRITE A
```

COBOL can support very rich data structures. A file suitable for Problem 5, called WORDS, with records, TRANS, that have sub-fields called ENGLISH and FRENCH each consisting of is alphabetic characters is established by

```
FD    WORDS
      DATA RECORD IS TRANS.
01    TRANS.
      02 ENGLISH PICTURE A(15).
      02 FRENCH PICTURE A(15).
```

A translation program then has the form

```
START.
      READ WORDS AT END GO TO FINISH.
      IF ENGLISH IS EQUAL TO "CHAIR"
      WRITE FRENCH.
      GO TO START.
FINISH.
```

PL1

The facilities possessed by PL/1 include a combination of those of FORTRAN and COBOL. As a general purpose language it is very complicated, and has not achieved its expected popularity. The slowness of its early implementations was a factor contributing to his. ALGOL 68, also a general purpose language, was adopted as a teaching language by many Computer Science departments because of the attractiveness of unified design based on a small number of independent concepts. However, PASCAL has tended to supersede it. The language itself is extensible in the sense that new features, such as operators and variable types can be defined and declared to suit the programmer. Implementation of the language in an entirely satisfactory manner has proved difficult.

Pascal

Pascal is descended from the ALGOLs. It was designed as a teaching language to demonstrate programming as a systematic discipline. It was also intended that it should be possible to implement the language compactly and efficiently. PASCAL appears likely to take over from BASIC as the most popular high-level language for microcomputers.

The computation and assignment instructions are similar to those of ALGOL 60. Repetition and output are illustrated by this solution to Problem 3 :

```
var n : integer ; n:= 10;
while n < 21 do
begin writeln (n); n:= n + 1 end
```

This solution to Problem 4 finds the largest of a set of positive numbers, the end of which is indicated by a negative number :

```
var a, b : real ; read (b) ; a:= b;
repeat
if b > a then a:= b ; read (b)
until b < 0 ;
writein (a)
```

(CHESS MACHINE CONGRESS)

Pascal provides all the control features necessary for structured programming. Arrays and complex data structures are also supported. A data structure suitable for Problem 5 can be declared in this way :

```
type wordpair = record
              english, french : packed away [1. .15] of char
              end;

var first : wordpair ;

first. english := "chair" ;
first. french := "chaise"
```

An array of variables of type wordpair can be declared, and then a solution to Problem 5 is straightforward.

LISP

LISP is a list processing language. The list is a useful representation in a variety of applications. For example, character strings may be regarded as lists of characters, and text as lists of the obstacles to its movement, it can determine whether a move it proposes to make is obstructed by scanning this list.

LISP is a functional language. Every instruction consists of a function and its arguments, and is executed by evaluating the arguments, applying the function to them and returning the resulting function value. Assignment is achieved by

```
(SETQ A 5)
```

which sets A to 5. Then

```
(PLUS A 6)
```

returns the value 11, because the arguments of PLUS are evaluated to 5 and 6, and applying the function PLUS to them gives 11. It should be noted that LISP programs are lists — the previous one is the list of the three elements PLUS, A and 6. Since programs and data have the same structure (both are lists) it is possible to write programs that compute other programs.

Problem 1 is solved in LISP by

```
(PLUS A (QUOTIENT (TIMES 2 B) C))
```

The programmer must put the arithmetic operations in the correct order.

More important than the arithmetic functions are the LISP functions for processing lists. These include CAR and CDR. After the assignment

```
(SETQ L '(A B C) )
```

which assigns the three-element list (A B C) to L, the instruction

```
(CAR L)
```

returns A, the first element of the list, while

```
(CDR L)
```

returns (B C), the list with its first element deleted.

A suitable data structure for Problem 5 is the list

```
((CHAIR CHAISE) (HOUSE MAISON) (HORSE CHEVAL))
```

if this list is assigned to L, then translation is achieved, essentially, by printing (CDR (CAR L)) when a match is found to (CAR (CAR L)).

Summary

Descriptions of a number of programming languages have been provided in this article. Even if your favourite language is not covered, it is undeniable that the languages mentioned here are popular. Merely to cover FORTRAN and COBOL would ensure that the languages in which the majority of programs are written are covered. As to the future, it is certain that new languages will be designed. Perhaps an ideal language that is all things to all men will emerge — could it be ADA?

Further Reading

Two books on programming languages each of which includes further references to individual languages are :

'An introduction to the study of programming languages', D.W. Barron, Cambridge University Press, 1977.

'A comparative study of programming languages', B. Higman, Macdonald and Jane's, 1967.

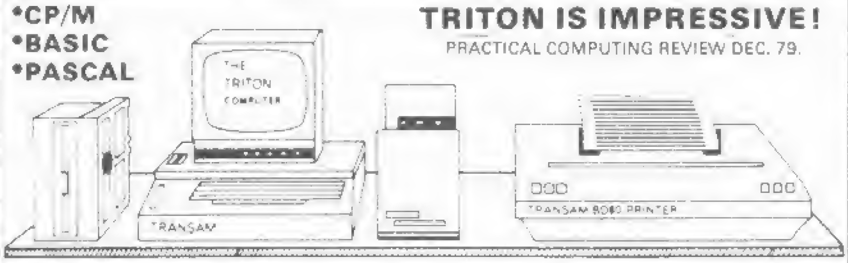
"WE WON'T GET MUCH OUT OF IT FOR THE NEXT FEW DAYS,— THIS IS A SICK NOTE!"

TRANSAM

COMPONENTS AND SYSTEMS FROM TRANSAM COMPUTERS

*CP/M
*BASIC
*PASCAL

TRITON IS IMPRESSIVE!
PRACTICAL COMPUTING REVIEW DEC. 79.



TRITON
COMPUTER SYSTEM.

Designed for ease of construction and flexibility. Kits come complete and all components and software are available separately. UK designed and supported. Fully documented hardware and software and a totally flexible approach to system building. Powerful and easy to use system monitors – a range of languages available. Firmware is Eprom based and upgrading from one level to the next is easy.

● L5.2 with 1.5k monitor 2.5k basic	£294.00
● L7.2 with 2k mon 8k extended basic	£409.00
● L8.2 4k ed/mon 20k res pascal	£611.00
● L9.2 CP/M disc based system	P.A.O.
● 8k ram card kit (21141)	£97.00
● 8k eprom cards (EXCL 8 x 2708)	£31.00
● Motherboard expansion 8 slot	£50.000
● Trap-res assm/edit, etc. (8 x 2708)	£80.00
● Transam BD80 bi-dir printer	£595.00
● TVM 10 video monitor 9"	£79.00
● Eprom prog (2708) kit	£29.50

SEND FOR OUR CATALOGUE FOR FULL DETAILS OF TRITON FEATURES!

CP/M AVAILABLE NOW FOR TRITON

Disc operating system complete with text editor, assembler, debugger, system utilities and complete file management. Makes Triton fully CP/M compatible and able to run CP/M based software. Triton will support up to four 5¼ or 8" drives single or double density full CP/M software user group facilities available. SAE for details.

CP/M Disk plus manuals (6) £75.00

DISK DRIVES & POWER SUPPLIES

SHUGART

SA400 5¼" drive	£205.00
SA800 8" drive	£380.00
Power one quality power supplies	
CP249 1 - 5¼ PSU	£33.00
CP323 2 - 5¼ PSU	£60.00
CP205 1 - 8" PSU	£56.00
CP206 2 - 8" PSU	£76.00

TCL PASCAL – CP/M compatible

A standard Pascal compiler available on a resident (20k) Eprom based configuration* or available to run under CP/M on 8" disc plus documentation. CP/M version £90.00.
*P.O.A.
TCL Pascal Manual and specification £6.50

FULL RANGE OF MICRO SUPPORT CHIPS – IN STOCK

SN74LS00N	22	SN74LS54N	21	SN74LS138N	95	SN74LS195AN	85	SN74LS325N	2.55	SUPPORT		RAMS	
SN74LS01N	22	SN74LS55N	21	SN74LS139N	95	SN74LS196N	1.20	SN74LS326N	2.56	8212	2.20	2101	2.32
SN74LS02N	26	SN74LS63N	1.50	SN74LS145N	1.20	SN74LS197N	1.20	SN74LS327N	2.55	8216	2.80	2102L 4	1.20
SN74LS03N	26	SN74LS73N	35	SN74LS148N	1.75	SN74LS221N	1.25	SN74LS352N	1.35	8224	2.80	2111	2.32
SN74LS04N	26	SN74LS74N	40	SN74LS151N	85	SN74LS240N	2.20	SN74LS353N	1.50	3853 (F8)	10.00	2112	2.46
SN74LS05N	26	SN74LS75N	48	SN74LS153N	80	SN74LS241N	1.90	SN74LS365N	65	8228	4.20	6810	4.00
SN74LS08N	20	SN74LS76N	35	SN74LS154N	1.60	SN74LS242N	1.90	SN74LS366N	85	8T26A	1.75	8154	11.50
SN74LS09N	22	SN74LS78N	35	SN74LS155N	1.25	SN74LS243N	1.95	SN74LS367N	65	8T28	1.90	2114L 450	6.50
SN74LS10N	18	SN74LS83AN	1.15	SN74LS156N	1.25	SN74LS244N	2.10	SN74LS368N	85	6522	8.75	2114L-250	7.60
SN74LS11N	26	SN74LS85N	1.10	SN74LS157N	80	SN74LS245N	3.60	SN74LS373N	1.75	8251	5.00	74C920	11.00
SN74LS12N	25	SN74LS86N	40	SN74LS158N	99	SN74LS247N	1.25	SN74LS374N	1.70	8253	11.00	74C921	11.00
SN74LS13N	55	SN74LS90N	65	SN74LS160N	1.15	SN74LS248N	1.95	SN74LS375N	72	8255	5.00	74C929	11.00
SN74LS14N	89	SN74LS91N	99	SN74LS161N	1.15	SN74LS249N	1.30	SN74LS377N	1.75	8257	£11.00	4027	5.00
SN74LS15N	25	SN74LS92N	90	SN74LS162N	1.15	SN74LS251N	1.45	SN74LS378N	1.32	8259	12.50	4044	7.00
SN74LS20N	20	SN74LS93BN	65	SN74LS163N	90	SN74LS253N	1.25	SN74LS379N	1.40	8155	12.50	4045	7.00
SN74LS21N	26	SN74LS95AN	1.20	SN74LS164N	1.60	SN74LS257N	1.40	SN74LS381N	3.65	6402	5.00	4060	7.00
SN74LS22N	26	SN74LS96N	1.75	SN74LS165N	1.70	SN74LS258N	95	SN74LS386N	57	6821P	4.60	2107	7.80
SN74LS26N	29	SN74LS107N	35	SN74LS166N	1.75	SN74LS259N	1.45	SN74LS390N	1.98	6850P	4.80	4116-58 (for 8)	8.00
SN74LS27N	36	SN74LS109N	39	SN74LS168N	1.95	SN74LS260N	39	SN74LS393N	1.50	6852P	5.50	4118	20.00
SN74LS28N	35	SN74LS112N	39	SN74LS169N	1.95	SN74LS261N	3.50	SN74LS395N	1.80	AY 5 2376	11.50	Z80P10	8.00
SN74LS30N	25	SN74LS113N	44	SN74LS170N	2.50	SN74LS266N	39	SN74LS396N	1.70	MC14411	12.00	Z80CTC	8.00
SN74LS32N	27	SN74LS114N	44	SN74LS173N	2.20	SN74LS273N	1.85	SN74LS398N	2.75	M57109	12.43	Z80AP10	9.50
SN74LS33N	39	SN74LS122N	79	SN74LS174N	1.15	SN74LS279N	79	SN74LS399N	1.60	M57160	10.00	Z80ACTC	9.50
SN74LS37N	29	SN74LS123N	90	SN74LS175N	1.06	SN74LS280N	1.75	SN74LS424N	4.50	M57161	10.00	EPROMS	
SN74LS38N	29	SN74LS124N	1.60	SN74LS181N	2.75	SN74LS283N	1.80	SN74LS445N	1.25	TMS6011	6.00	1702	5.00
SN74LS40N	25	SN74LS125N	65	SN74LS190N	1.75	SN74LS290N	1.80	SN74LS447N	1.25	81LS95	1.80	5204	5.00
SN74LS42N	79	SN74LS126N	65	SN74LS191N	1.75	SN74LS293N	1.80	SN74LS490N	1.95	81LS96	1.80	2708	8.00
SN74LS47N	95	SN74LS132N	75	SN74LS192N	1.45	SN74LS295AN	2.20	SN74LS668N	95	81LS97	1.80	2516	25.00
SN74LS48N	95	SN74LS133N	35	SN74LS193N	1.75	SN74LS298N	2.20	SN74LS669N	95	81LS98	1.80	2532	50.00
SN74LS49N	1.09	SN74LS136N	40	SN74LS194AN	1.89	SN74LS324N	1.80	SN74LS670N	2.70				

DIL PLUG SOCKETS & SWITCHES

W/WRAP SKTS		DIL SKTS		DIL PLUGS		DIL SWITCHES	
8DIL	0.20	8DIL	0.14	14DIL	0.40	4DIL	1.20
14DIL	0.35	14DIL	0.15	16DIL	0.45	7DIL	1.75
16DIL	0.42	16DIL	0.17	SCOTCHFLEX		8DIL	1.80
18DIL	0.60	18DIL	0.24	14DIL	1.30	16w ZIF*	4.95
24DIL	0.52	20DIL	0.27	16DIL	1.50	24w ZIF*	6.20
28DIL	0.74	24DIL	0.30	24DIL	2.80		
40DIL	0.95	28DIL	0.36				
		48DIL	0.50				

*ZERO INSERTION FORCE

COMPUCOLOR II – FULL COLOUR

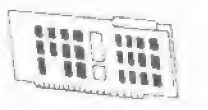
- 13" 8 colour crt display
- Built in 5¼ disk
- 16k extended basic in rom
- 71 key keyboard - detached
- RS232 - 50 pin bus
- 8k user RAM – fully expandable

SPECIAL GRAPHICS PACKAGE

£985 Plus VAT

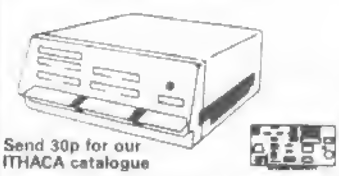
S100 DISC CONTROLLER DOUBLE DENSITY £195 Plus VAT

As used on Triton. Fully built will drive 8 - 8" or 8 - 5¼ drives. Single or double sensity. Works with all Shugart compatible drives. Uses the 1791 chip on board crystal – CPU independent



DPS.1 MAINFRAME – PASCAL SYSTEM

S100 to IEEE spec



Send 30p for our ITHACA catalogue

ITHACA

PASCAL/Z build your own Pascal Micro Development system. IEE-S100 bus system using DPS1 main-frame. Supports K2, ASSEMBLE/Z and PASCAL/Z on 8" disc
Complete system £2910.00

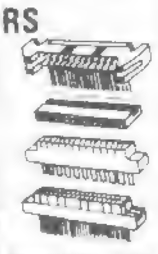
WE STOCK THE FULL RANGE OF S100 CARDS AND ACCESSORIES

S100 BOARDS

8k Static RAM board (450ns)	£99.00
8k Static RAM board (250ns)	£117.00
Z80 cpu board (2MHz)	£106.00
Z80 cpu board (4MHz)	£123.00
2708/27 16 EPROM board	£57.00
Prototype board (bare board)	£15.00
Video display board (64 x 16, 128U/L Ascu)	£106.75
Disc controller board	£131.25
K2 disc operating system	£45.00
ASSEMBLE/Z Macro Assm	£37.50
PASCAL/Z compiler	£205.00
PASCAL/Z CP/M	£235.00
16k Static RAM	£275.00

MULTIWAY CONNECTORS

INSULATION PIERCING			
20 way plug	2.30	35/70	4.80
26 way plug	2.70	36/72	4.74
34 way plug	3.30	40/80	5.00
50 way plug	4.60	43/86	5.50
20 way skt	3.40	50/100	5.80
26 way skt	4.00	GOLD .156 PITCH	
34 way skt	4.80	6/12	1.25
30 way skt	6.00	10/20	1.50
EDGE CONN PCB		12/24	2.00
GOLD .1" PITCH		15/30	2.20
		18/36	2.30
22/44	3.20	22/44	2.65
25/50	3.60	28/56	3.30
28/56	3.90	36/72	3.90
30/60	4.15	43/86	4.60

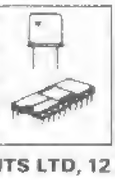


64 way DIN male	2.50
64 way DIN female	4.50

VISIT OUR SHOWROOM

WE ALSO STOCK:– A comprehensive range of books and magazines, VERO products including S100 and Eurocard and Wire Wrap equipment, Weller soldering equipment, Ribbon Cables, tools, tapes, diskettes, connectors and OK Tool range.
Systems continuously on display in our showroom.

CRYSTALS					
100k	3.00	4MHz	2.10	FB (3850)	9.50
200k	3.70	4.43M	1.00	8080A	8.33
1MHz	3.60	5MHz	2.70	6809	24.00
1008k	3.50	6MHz	2.70	Z80	8.00
1843k	3.00	7MHz	2.70	Z80A	15.00
2MHz	1.50	7.168M	2.50	8085A	12.95
2457k	3.05	8MHz	2.70	6502	8.00
3276k	2.70	10MHz	2.70	SCMP II	10.00
		10.7M	2.70	6802	13.95



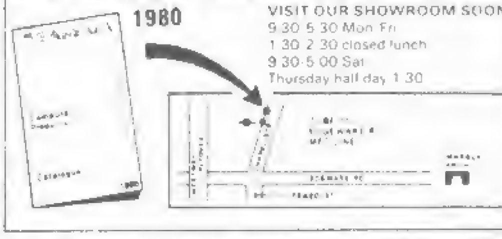
ALL PRICES
Exclude VAT & P/P
VAT 15% P & P 40p on small orders
For larger items please Tel
Telephone credit card orders accepted subject to £5 min
RAPID MAIL ORDER SERVICE

CATALOGUE

NEW A4 SIZED ONLY 40p & S.A.E

1980

VISIT OUR SHOWROOM SOON
9.30-5.30 Mon-Fri
1.30-2.30 closed lunch
9.30-5.00 Sat
Thursday half day 1.30



TRANSAM COMPONENTS LTD, 12 CHAPEL STREET, LONDON NW1 Tel: 01-402 8137 Telex: 444898

INSTANT SOFTWARE

CT Software is a unique service that we offer to our readers. Each program comes on a high quality tape packaged with full documentation.

Read down the list of titles—you'll find a few suprises and some things for which you have offered sacrifices on a stone at dawn before now. All are checked and fully quaranteed, any complaints and we'll replace by return of post.

We think this is a revolution in reader service and one that will change the way you use your computer. So why not try us out ?

All orders and enquiries to:— CT software,
4 Morgan Street,
London E3 5AB.

All programs except 0013R are £6.75.

TRS 80 Level 1

BUSINESS PACKAGE I Keep the books for a small business with your TRS-80 Level I 4K. The six programs included are:

General Information — The instructions for using the package.

Fixed Asset Control — This will give you a list of your fixed assets and term depreciation.

Detail Input — This program lets you create and record your general ledger on tape for fast access.

Month and Year to Date Merge — This program will take your monthly ledger data and give you a year to date ledger.

Profit and Loss — With this program you can quickly get trial balance and profit and loss statements.

Year End Balance — This program will combine all your data from the profit and loss statements into a year end balance sheet.

With this package, you can make your TRS-80 a working partner.

Order Code. 0017R

PERSONAL FINANCE I Let your TRS-80 handle all the tedious details the next time you figure your finances:

Personal Finance I — With this program you can control your incoming and outgoing expenses.

Checkbook — Your TRS-80 can balance your checkbook and keep a detailed list of expenses for tax time.

This handy financial control package for the home requires only a TRS-80 Level I 4K. Order No. 0027R 5.75.

Level 1&2

AIR FLIGHT SIMULATION Turn your TRS-80 into an airplane. You can practice takeoffs and landings wiht the benefit of full instrumentation. This one-player simulation requires a TRS-80 Level I 4K, Level II 16K. Order No. 0002R

SPACE TREK II Protect the quadrant from the invading Klingon warships. The Enterprise is equipped with phasers, photon torpedoes, impulse power, and warp drive. It's you alone and your TRS-80 Level I 4K, Level II 16K against the enemy. Order No. 0002R

SANTA PARAVIA AND FIUMACCIO Become the ruler of a medieval city-state as you struggle to create a kingdom. Up to six players can compete to see who will become the King or Queen first. This program requires a 16K TRS-80 Level I & II. Order No. 0043R

ELECTRONICS I This package will not only calculate the component values for you, but will also draw a schematic diagram, too. You'll need a TRS-80 Level I 4K, Level II 16K to use:

Tuned Circuits and Coil Winding — Design tuned circuits without resorting to cumbersome tables and calculations.

555 Timer Circuits — Quickly design astable or monostable timing circuits using this popular IC.

LM 381 Preamp Design — Design IC preamps with this low-noise integrated circuit.

This package will reduce your designing time and let you build those circuits fast. Order No. 0008R

HAM PACKAGE I This versatile package lets you solve many of the commonly encountered problems in electtronics design. With your Level I 4K or Level II 16K TRS-80, you have a choice of:

Basic Electronics with Voltage Divider — Solve problems involving Ohm's Law, voltage dividers, and RC time constants.

Dipole and Yagi Antennas — Design antennas easily, without tedious calculations. This is the perfect package for any ham or technician. Order No. 0007R

Level 2

TRS-80 UTILITY I Ever wonder how some programmers give their programs that professional look? Instant Software has the answer with the TRS-80 Utility I package. Included are:

RENUM — Now you can easily renumber any Level II program to make room for modification, or to clean up the listing.

DUPLIK — This program will let you duplicate any BASIC, assembler, or machine-language program, verify the data, merge two or more programs into one data block, and even copy Level I programs on a Level II machine. For TRS-80 Level II 16K. Order No. 0081R

TRS-80 UTILITY 2 Let Instant Software change the drudgery of editing your programs

into a quick, easy job. Included in this package are:

CFETCH — Search through any Level II program tape and get the file names for all the programs. You can also merge BASIC programs, with consecutive line numbers, into one program.

CWRITE — Combine subroutines, that work in different memory locations into one program. This works with BASIC or machine-language programs and gives you a general checksum.

This package is just the thing for your TRS-80 Level II 16K. Order No. 0076R

SPACE TREK IV Trade or wage war on a planetary scale. This package includes:

Stellar Wars — Engage and destroy Tie fighters in your attack on the Death Star. For one player.

Population Simulation — A two-player game where you control the economy of two neighbouring planets.

You decide, guns or butter, with your TRS-80 Level II 16K. Order No. 0034R

RAMROM PATROL/TIE FIGHTER/KLINGON CAPTURE Buck Rogers never had it so good. Engage in extraterrestrial warfare with:

Ramrom Patrol — Destroy the Ramron ships before they capture you.

Tie Fighter — Destroy the enemey Tie fighters and become a hero of the rebellion.

Klingon Capture — You must capture the Klingon ship intact. It's you and your TRS-80 Level II 16K battling across the galaxy. Order No. 0028R

CARDS This one-player package will let you play cards with your TRS-80 — talk about a poker face!

Draw and Stud Poker — These two programs will keep your game sharp.

No-Trump Bridge — Play this popular game with your computer and develop your strategy.

This package's name says it all. Requires a TRS-80 Level II 16K. Order No. 0063R

HOUSEHOLD ACCOUNTANT Let your TRS-80 help you out with many of your daily household calculations. Save time and money with these fine programs:

Budget and Expense Analysis — You can change budgeting into a more pleasant job with this program. With nine sections for income and expenses and the option for one- and three-month review or year totals, you can see where your money is going.

Life Insurance Cost Comparison — Compare the cost of various life insurance policies. Find out the difference in price between term and whole life. This program can store and display up to six different results.

Datebook — Record all those important dates in your life for fast, easy access. The program has all major holidays already included.

All you need is TRS-80 Level II 16K. Order No. 0069R

FINANCIAL ASSISTANT Compute the figures for a wide variety of business needs. Included are:

Depreciation — This program lets you figure depreciation on equipment in five different ways.

Loan Amortization Schedule — Merely enter a few essential factors, and your TRS-80 will display a complete breakdown of all costs and schedules of payment for any loan.

Financier — This program performs thirteen common financial calculations. Easily handles calculations on investments, depreciation, and loans.

1% Forecasting — Use this simple program to forecast sales, expenses, or any other historical data series.

All you need is a TRS-80 Level II 16K. Order No. 0072R

PET

CASINO I These two programs are so good, you can use them to check out and debug your own gambling system!

Roulette — Pick your number and place your bet with the computer version of this casino game. For one player.

Blackjack — Try out this version of the popular card game before you go out and risk your money on your own "surefire" system. For one player.

This package requires a PET with 8K. Order No. 0014P

CASINO II This craps program is so good, it's the next best thing to being in Las Vegas or Atlantic City. It will not only play the game with you, but also will teach you how to play the odds and make the best bets. A one player game, it requires a PET 8K. Order No. 0015P

CHECKERS/BACCARAT Play two old favourites with your PET.

Checkers — Let your PET be your ever-ready opponent in this computer-based checkers program.

Baccarat — You have both Casino- and Blackjack-style games in this realistic program.

Your PET with 8K will offer challenging play anytime you want. Order No. 0022P

MIMIC Test your memory and reflexes with the five different versions of this game. You must match the sequence and location of signals displayed by your PET. This one-player program includes optional sound effects with the PET 8K. Order No. 0039P

TREK-X Command the Enterprise as you scour the quadrant for enemy warships. This package not only has superb graphics, but also includes programming for optional sound effects. A one-player game for the PET 8K. Order No. 0032P

TURF AND TARGET Whether on the field or in the air, you'll have fun with Turf and Target package. Included are:

Quarterback — You're the quarterback as you try to get the pigskin over the goal line. You can pass, punt, hand off, and see the results of your play using the PET's superb graphics.

Soccer II — Play the fast-action game of soccer with four playing options. The computer can play itself, play a single player, two players with computer assistance, and two players without help.

Shoot — You're the hunter as you try to shoot the bird out of the air. The PET will keep score.

Target — Use the numeric keypad to shoot your puck into the hom position as fast as you can.

To run and score all you'll need is a PET with 8K. Order No. 0097P

ARCADE I This package combines an exciting outdoors sport with one of America's most popular indoor sports:

Kite Fight — It's a national sport in India. After you and a friend have spent several hours manoeuvering your kites across the screen of your PET, you'll know why!

Pinball — By far the finest use of the PET's exceptional graphics capabilities we've ever seen, and a heck of a lot of fun to play to boot.

Requires an 8K PET. Order No. 0074P

ARCADE II One challenging memory game and two fast-paced action games make this one package the whole family will enjoy for some time to come. Package includes:

UFO — Catch the elusive UFO before it hits the ground!

Hit — Better than a skeet shoot. The target remains stationary, but you're moving all over the place.

Blockade — A two-player game that combines strategy and fast reflexes.

Requires 8K PET. Order No. 0045P

DUNGEON OF DEATH Battle evil demons, cast magic spells, and accumulate great wealth as you search for the Holy Grail. You'll have to descend into the Dungeon of Death and grope through the suffocating darkness. If you survive, glory and treasure are yours. For the PET 8K. Order No. 0064P

Apple

MATH TUTOR I Parents, teachers, students, now you can turn your Apple computer into a mathematics tutor. Your children or students can begin to enjoy their math lessons with these programs:

Hanging — Perfect your skill with decimal numbers while you try to cheat the hangman.

Spellbinder — Cast spells against a competing magician as you practice working with fractions.

Whole Space — While you exercise your skill at using whole numbers your ship attacks the enemy planet and destroys alien spacecraft.

All programs have varying levels of difficulty. All you need is Applesoft II with your Apple II 24K. Order No. 0073A

MATH TUTOR II Your Apple computer can go beyond game playing and become a mathematics tutor for your children. Using the technique of immediate positive reinforcement, you can make math fun with:

Car Jump — Reinforce the concept of calculating area while having fun making your car jump over the ramps.

Robot Duel — Practice figuring volumes of various containers while your robot fights against the computer's mechanical man.

Sub Attack — Take the mystery out of working with percentages as your submarine sneaks into the harbor and destroys the enemy fleet.

All you need is Applesoft II with your Apple II and 20K. Order No. 0098A

GOLF Without leaving the comfort of your chair, you can enjoy a computerized 18 holes of golf with a complete choice of clubs and shooting angles. You need never cancel this game because of rain. One or two players can enjoy this game on the Apple with Applesoft II and 20K. Order No. 0018A

BOWLING/TRILOGY Enjoy two of America's favorite games transformed into programs for your Apple:

Bowling — Up to four players can bowl while the Apple sets up the pins and keeps score. Requires Applesoft II.

Trilogy — This program can be anything from a simple game of tic-tac-toe to an exercise in deductive logic. For one player.

This fun-filled package requires an Apple with 20K. Order No. 0040A

CLASSIFIED INFORMATION

Semi-Display:–
1- 3 insertions — £5.00 per single column centimetre
4-11 insertions — £4.50 per s.c.c.
12 insertions — £4.00 per s.c.c.

Classified:–
19 pence per word (minimum 25 words) · Box number on application (Personal ads only) £1.00 extra to cover 5 replies.

ALL ADVERTISEMENTS IN THIS SECTION MUST BE PRE-PAID

Closing date:— 2nd Friday in month preceding publication. Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request). Cheques and postal orders should be crossed and made payable to 'Computing Today'.

CLASSIFIED ADS, COMPUTING TODAY, 145 CHARING CROSS ROAD, LONDON WC2H 0EE (Tel. 01-437 1002)

CLASSIFIED

TRITON, cased, full board RAM, LS.1 Basic and Monitor, £250, or will consider it part exchange for TRS 80, 16K, Level 2. Must include video display, tape recorder. Boyd, 1 Church Close, Blackfordby, Burton-on-Trent, Staffs. DE11 8AT.

MK14 CORNER Interface Board, includes Flag Driven Mains Relay, LED Indicators for all Serial I/O, A/D and Single Step Chip, Prototype Area; PCB and Circuit £3.95. Replace calculator display with ½" FND500s; PCB, Filter and Instructions £1.95. Ready-built Replacement Keyboard £11. Useful notes on MK14 75p. Programming sheets 95p a pad. Rayner, 'Kismet', High Street, Colnbrook, Bucks.

NASCOM I built seen working B. Bug. Tape Recorder various programmes full Documentation plus set Z80 manuals £110 Liverpool 051 428 3636 evenings.

FOR SALE: MK14, Ram 1/0, Extra Ram, Cassette Interface, Modified keyboard, VDU, PSU complete all working £75 – Brighouse 710498.

BARGAIN. Challenger IP plus cassette recorder plus software tapes plus manuals. All as new. 8K RAM, 8K Microsoft Basic. UHF or Video Output. £250 o.n.o. Ruislip 72852 after 6.30 p.m.

ACTION PROGRAMS FOR NASCOM. 1 (1) Dice Pontoon runs on unexpanded Nascom with T2/T4 or B-Bug. (2) Breakout runs on expanded Nascom 1 with 8k Tape Basic and T2/T4 or B-Bug. Above programs £3.50 each inclusive. T. Edwards, 70 Waterhouse Moor, Harlow, Essex.

NEED HARD COPY? IBM GOLFBALL input/output writer £75. R0390 Ascii printer £75. 1000 Char. VDU £50. 01-778-3600 Evenings.

NASCOM 1, boxed, with PSU, auto cassette, extra serial output for printer, masses of programs. £150 ono. Tel. Macclesfield (0625) 32203.

PET 32K, £695.00, with new improved keyboard and complete with external cassette deck. Special Low Price £695.00 including V.A.T. and free dust cover. Contact MICHAEL PRICE 19 Wise Grove, Warwick, Warwickshire. Tel. after 6pm 0926-86-2380 will deliver.

FASTER TYPING ON COMUKIT. Basic keywords in one touch. Control-I and INPUT appears on screen; Control-T equals TAB etc. £2.95 plus SAE. A. R. Leeder, Haileybury, Hertford SG13 7NU.

NASCOM 1 SOUND, bring your programs to life. Complete kit of parts for sound output buffer. No on-board modifications, includes "Alien attack" or "Parachute drop" game program listing with sound subroutine to add to your own programs. £7.00. Also available "Copycat" (number game), "Number base conversion" (programming aid), "Cowboy" (shooting game for systems with special graphics only). All fully commented listings, not suitable for nas-sys. £4.50 each. SAE further details. Ramon Electronics, 94 Linden Crescent, Folkestone, Kent.

PET 2001-N 16K, plus external cassette, a few programmes. As new, only 3 months old, £650 ono. Tel. 0865-820632 Evenings.

UK 101 SOFTWARE (ALSO SUPERBOARD II). Startrek, Spaceinvaders, etc., £3 each. All Software requirements considered. UK 101 built and tested, £250 inclusive. T. W. Software, 34 Hillier Close, Barnet, Herts.

PLAY SPACE INVADERS on your NASCOM. Full feature, 3K object code. Many options. SAE for full details. J. Atkins, 37 Wellington Road, Maidenhead, Berks. Telephone 0628-35145.

NASCOM I COMPUTER SYSTEM, with T2 monitor and documentation, can be seen working, £130 ono with P.S.U. Phone: Leicester 413796 after 4 p.m.

NASCOM PROGRAMS in machine code. 20 useful assorted programs to run under T2, B-Bug, T4 or Nas-sys, £3 or send SAE for further details. L. Sargent, 134 Stockham Park, Wantage, Oxon.

PET 2001-8K, excellent condition, complete with software, £395 ono. Also Pet 3022 and Line Printer with traction feed, only one month old with cable, £500 ono. Crowthorne 5594.

ADVERTISERS INDEX

BARINGLOCK LTD 51
B.N.R.S. 18
BUSINESS & LEISURE 18
CAMBRIDGE LEARNING ENTERPRISES 54
CATER KEYBOARDS 4
CHROMASONICS 18
COMMODORE 13
COMP, COMP, COMP . . . . 82 & 83
GORGRANGE COMMUNICATORS 66
HAPPY MEMORIES 4
A. J. HARDING (MOLIMERX) 4
HENRY'S RADIO 5
INTERFACE COMPONENTS 34
LOWE ELECTRONICS 17
MASTERPACK 65
MIGHTY MICRO 30
NASCOM 67
NEWBEAR 84
NEWTRONICS 20
NIC MODELS 81
PETSOFT 8
POWERTRAN COMPUTERS 2
SCIENCE OF CAMBRIDGE . . . . 70 & 71
SGS ATES 19
SOFTWARE PUBLISHING CO 51
TANGERINE 10
TIMEDATA 51
TRANSAM COMPONENTS 77
WILLIAM STUART SYSTEMS 4
WINCHESTER TECHNOLOGY 51

8MHz Super Quality Modulators	**£4.90**
6MHz Standard Modulators	**£2.90**
C12 Computer Grade Cassettes 10 for	**£4.00**
Super Multi-rail P.S.U. +5 −5 +12v	**£29.50**
ETI Breakout Game — Chip and PCB	**£9.90**
S100 Expansion Motherboard for Nascom I	**£39.00**
Anadex Printer Paper — 2000 sheets	**£25.00**
Floppy Disks 5¼" Hard & Soft Sectored	**£3.50**
Floppy Disk Library Case 5¼"	**£3.50**
Eprom Boards	**£63.00**
8K Static Ram Boards — S100	**£110.00**
Cartridges for Grandstand	**£11.99**
George Risk Ascii Keyboard	**£39.00**
Cartridges for Atari — Full Range in Stock	**£13.90**
Interface PET IEEE — Centronics Parallel Not decoded **£49.00** Decoded	**£77.00**
Interface to Centronics parallel for TRS80	**£75.00**
Verocases for Nascom 1 & 2 etc.	**£22.50**
Keyboard Cases	**£9.90**
Electric Pencil for TRS80	**£29.00**

RRP £690 only **£590** + VAT

IBM SELECTRIC GOLFBALL

Refurbished IBM Golfball Printers to new specs. Accepts Centronics parallel data. Friction feed. Prints at 15 cps.

Attaches either directly or through interfaces to Pet, Apple, TRS80, Sorcerer, Nascom, Compukit etc.

HITACHI PROFESSIONAL MONITORS

9" — **£129**
12" — **£199**

● **Reliability** Solid state circuitry using an IC and silicon transistors ensures high reliability. ● **500 lines horizontal resolution** Horizontal resolution in excess of 500 lines is achieved in picture center. ● **Stable picture** Even played back pictures of VTR can be displayed without jittering. ● **Looping video input** Video input can be looped through with built-in termination switch. ● **External sync operation** (available as option for U and C types) ● **Compact construction** Two monitors are mountable side by side in a standard 19-inch rack.

only **£356** + VAT

TRS80 LEVEL 2 16K

Fully converted to UK T.V. Standard. Comes complete with easy to follow manuals. UK Power Supply — Cassette Leads — Sample tapes. Special box to enable you to plug into your own TV. Recommended for first time-buyers. Just plug in and go. **Full Range of Software Available**
Model with numeric key pad £389 + VAT
4K Level I — machine only £251 + VAT

NEW REDUCED PRICES

8K **£449**
16K **£549** + VAT
32K **£649**

RRP £795 for 32K

The PEDIGREE PETS

Very popular for home & business use. 8K Microsoft Basic in ROM. 8K Pet 32K & 16K with new improved keyboard. All with green screen.
Extra cassette deck **£55** Full range of software available.

32K **£690** + VAT

48K **£790** + VAT

EXIDY SORCERER

RRP £859 for 32K

For Personal or Business Use.
32K or 48K memory. 8K Microsoft Basic in ROM. Dual Cassette I/O, RS232 I/O, Parallel I/O (Centronics). Expansion available through optional extra S100 Motherboard. 69 Key keyboard including 16 key numeric pad.

S100 EXPANSION — **£199**

NASCOM IMP PLAIN PAPER PRINTER

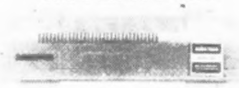
Fully built and housed in a stylish enclosure for just **£325** plus VAT. Interfaces with all micro computers.

The Nascom IMP (Impact Matrix Printer) features are listed below:

● 60 lines per minute. ● 80 characters per line. ● Bi-directional printing. ● 10 line print buffer. ● Automatic CR/LF. ● 96 character ASCII set (including upper/lower case, $, #, £). ● Accepts 8½" paper (pressure feed). ● Accepts 9½" paper (tractor feed). ● Tractor/pressure feed. ● Baud rate from 110 to 9600. ● External signal for optional synchronisation of baud rate. ● Serial RS232 interface with parallel option available soon.

only **£295** + VAT

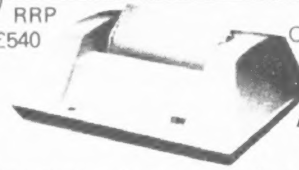
32K TRS80 EXPANSION INTERFACE



Expand your TRS80 by 32K.
32K Memory on board. Centronics parallel port. Disk controller card. Real time clock. Requires Level II Basic. Interface for 2 cassette decks, complete with power supply.

RRP £540

only **£499** + VAT



ANADEX DP8000

Super Quality — Low cost printer. Tractor Feed with full 96 ASCII character set. Accepts RS232C at band rates between 100 and 9600 and Parallel Bit data.
Attaches either directly or through interfaces to Pet, Apple, TRS80, Sorcerer, Nascom, Compukit etc.

video 100

12" BLACK & WHITE LOW COST VIDEO MONITOR

RRP £79

only **£69** + VAT

● Ideal for home, personal and business computer systems ● 12" diagonal video monitor ● Composite video input ● Composite video input ● Compatible with many computer systems ● Solid-state circuitry for a stable & sharp picture ● Video bandwidth - 12MHz + 3DB ● Input impedance -75 Ohms ● Resolution - 650 lines Minimum In Central 80% of CRT; 550 Lines Minimum beyond central 80%.

NASCOM-2 MICRO-COMPUTER

only **£295** + VAT

Your choice of freebies with every Nascom 2 purchased from us

either FREE POWER SUPPLY
OR FREE GRAPHICS ROM
OR FREE VERO CASE TO TAKE NASCOM 2

Microprocessors Z80A 8 bit CPU. This will run at 4MHz but is selectable between 1/2/4 MHz. This CPU has now been generally accepted as the most powerful, 8 bit processor on the market.

INTERFACE

Keyboard New expanded 57 key Licon solid state keyboard especially built for Nascom. Uses standard Nascom monitor controlled, decoding.

T.V. The 1v peak to peak video signal can drive a monitor directly and is also fed to the on-board modulator to drive the domestic T.V.

I.O. On-board UART (Int 6402) which provides serial handling for the on-board cassette interface or the RS232/20mA teletype interface.
The cassette interface is Kansas City standard at either 300 or 1200 baud. This is a link option on the NASCOM-2.
The RS232 and 20mA loop connector will interface directly into any standard teletype.
The input and output sides of the UART are independently switchable between any of the options —
i.e. it is possible to house input on the cassette and output on the printer.

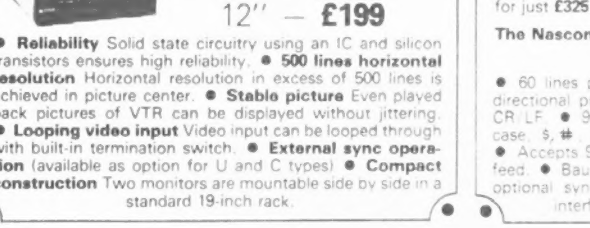
PIO There is also a totally uncommitted Parallel I/O (MK3881) giving 16, programmable, I/O lines. These are addressable as 2 x 8 bit ports with complete handshake controls.

Documentation Full construction article is provided for those who buy a kit and an extensive software manual is provided for the monitor and Basic.

Basic The Nascom 2 contains a full 8K Microsoft Basic in one ROM chip with additional features like DEEK, DOKE, SET, RESET for simple programming.
With free 16K RAM board.

A PROFESSIONAL WORD PROCESSING SYSTEM AND IT'S A COMPUTER AS WELL.

FOR ONLY
£3250 + VAT



PET 32K — This is the standard 32K Pet from Commodore. Reverse video and graphics allow the WordPro Package to give simple clear and easy to read displays.

2040 Disk Drives Twin disk drives allow large high speed storage for your letters, or paragraphs. Plugs in the back of the PET.

NEC Spinwriter NEC's high quality printer uses a print "thimble" that has less diameter and inertia than a daisy wheel, giving a quieter, faster, more reliable printer that can cope with plotting and printing (128 ASCII characters) with up to five copies, friction or tractor fed. The ribbon and thimble can be changed in seconds.
55 characters per second bidirectional printing — with red/black, bold, subscript, superscript, proportional spacing, tabbing, and much, much more.

WordPro II The heart of the system — consists of a ROM and diskette. The ROM is inserted into a space socket inside the Pet. One of the most versatile Word Processing Packages around.

All items sold separately.